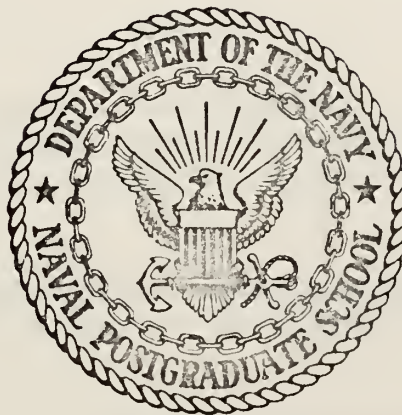


TOWARDS AN ENGLISH LANGUAGE
INTERACTIVE SIMULATION SYSTEM

Joseph Leon Clapper

NAVAL POSTGRADUATE SCHOOL

Monterey, California



THESIS

TOWARDS AN ENGLISH LANGUAGE
INTERACTIVE SIMULATION SYSTEM

by

Joseph Leon Clapper

Kenneth Stanley Nelson

Thesis Advisor:

G. E. Heidorn

December 1972

Towards an English Language
Interactive Simulation System

by

Joseph Leon Clapper
Lieutenant, United States Naval Reserve
B.S., Pennsylvania State University, 1968

and

Kenneth Stanley Nelson
Lieutenant, United States Navy
B.S., Oregon State University, 1966

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December 1972

ABSTRACT

Research at the Naval Postgraduate School has led to the development of a system for producing GPSS simulation programs for simple queuing problems through English language dialogue with an IBM 360/67 computer. This thesis describes work done to give the system the capability to actually perform the simulation and report the results through English language dialogue. A complete sample terminal session is included.

TABLE OF CONTENTS

I.	INTRODUCTION-----	6
	A. BACKGROUND-----	7
	B. THESIS OBJECTIVE-----	9
	C. ORGANIZATION OF THE THESIS-----	9
II.	DESCRIPTION OF NLP AND NLPQ-----	10
	A. BASIC CONCEPTS-----	10
	B. THE INTERNAL PROBLEM DESCRIPTION-----	12
	C. DECODING THE PROBLEM DESCRIPTION-----	14
	D. ENCODING FROM THE IPD-----	16
	E. THE SIMULATION ROUTINE-----	18
III.	A SAMPLE SIMULATION PROBLEM-----	21
IV.	IMPLEMENTATION-----	32
	A. FORTRAN ROUTINES AND MODIFICATIONS-----	32
	1. NLP Modifications to Allow X-Vector Read and Write-----	32
	2. Establishing a Linkage for Communication-----	33
	3. Modified Output Routine-----	34
	4. The Simulation Subroutine-----	37
	a. SIMULT-----	37
	b. SIMOUT-----	41
	c. SETIND-----	46
	d. SPSTAT-----	47

B. RULE ADDITIONS AND MODIFICATIONS-----	48
1. Named Records -----	49
2. Simulation Control Commands -----	49
3. Producing Selective Simulation Results -----	51
4. Interrogating the Simulation Results -----	53
V. CONCLUSIONS AND RECOMMENDATIONS -----	56
APPENDIX A FORTRAN Additions and Alterations-----	58
APPENDIX B Dictionary of Simulation Routine Variables -----	60
APPENDIX C Simulation Routine Listing-----	70
APPENDIX D NLPQ Rule Additions and Modifications -----	104
LIST OF REFERENCES -----	108
INITIAL DISTRIBUTION LIST -----	109
FORM DD 1473 -----	110

LIST OF FIGURES

1.	Internal Structure of the X-Vector -----	19
2.	Numerical Evaluation for OUTPUT Segment -----	36
3.	Statistic Switch Vector -----	43

I. INTRODUCTION

Since the advent of the computer, men have been striving to find easier methods of communicating their problems to the machine. Primitive communication was established by using the language of the machine and conversing at the computer's elementary level. In an effort to narrow the communication gap between man and machine, translators have been developed. These translators - assemblers and compilers - facilitate communication with the computer at a level considerably removed from the machine's language. However, even at this level, man is required to learn a new language in order to converse with his powerful assistant. Although significant advances have been made toward generalized, multi-purpose, higher level languages, considerable effort must be expended to learn and utilize these languages in a problem solving situation. A desirable alternative would be to have the ability to specify the problem directly to the machine in a natural language such as English, and have the machine solve the problem and report the results as requested by the user.

Several research projects have investigated various aspects of natural language interaction between man and computer [1,2]. Developments in the fields of linguistics and artificial intelligence have served to provide basic conceptual structures for natural language processing. One such development is the theory of stratificational linguistics

proposed by S. M. Lamb [3]. In this theory, language is considered to be a multi-level system of relationships.

A research project is currently being conducted at the Naval Postgraduate School to investigate using natural language man-machine interaction in solving queuing problems by simulation [4]. The system being developed is called NLPQ and is a specific application of a more general system called NLP. This work is based on the concepts of stratificational linguistics and utilizes an entity-attribute-value data structure. Background information on the development of both systems is included in this chapter. A further description of each system is included in Chapter II.

A. BACKGROUND

The initial work done on the project was the development of the general system NLP or Natural Language Processor. The constituents of this system are a "rule language" and a set of FORTRAN routines which compile and execute statements in the rule language. System monitor functions are also performed by the main routine. The system is implemented on the IBM 360/67 and is executed under control of the CP/CMS time sharing system.

With the basic system established, further research began to produce the rule modules necessary to handle a queuing system application (NLPQ). The form of an internal problem description (IPD) for a queuing problem was decided upon and a set of encoding rules

were written to convert the information contained in an IPD to a GPSS program [5]. Additional encoding rules were added to produce an English text description of the information contained in an IPD [6]. The generality of the English encoding rules also permits their use in other areas involving natural language responses from the computer. A set of English decoding rules was then developed to allow the user to describe his queuing problem to the computer in English. These rules perform the function of processing the input English text to produce the IPD. In addition, they are utilized in handling English language requests from the user.

Further research developed modules which would massage and inspect the IPD for missing or erroneous information before producing a GPSS program [7,8]. In cases where missing or erroneous information is detected, a request is made to the user to supply or correct the required information. A practical by-product of this research is the capability to enter an English problem description in a question-answer mode.

More recently, a FORTRAN subroutine designed to perform a GPSS-like simulation and a set of encoding rules to initialize the data structure required by this routine were developed [9,10]. Information contained in the IPD can be manipulated by these rules to produce either a GPSS program or a representation of the queuing problem in the data structure utilized by the simulation routine, or both.

B. THESIS OBJECTIVE

The objective of the research for this thesis was to integrate the simulation routine and associated rules into the existing NLPQ system to produce an initial version of an interactive simulation system.

This required making modifications and extensions to both the simulation routine and several of the existing rule modules.

C. ORGANIZATION OF THE THESIS

Chapter II of this thesis presents a more detailed discussion of pertinent portions of NLP and NLPQ. Chapter III contains a sample session to illustrate the capabilities of NLPQ in an interactive problem solving situation. The considerations involved in the implementation of the interactive simulation capability are discussed in detail in Chapter IV. Finally, Chapter V presents conclusions and recommendations for further research.

II. DESCRIPTION OF NLP AND NLPQ

Since the interactive simulation capabilities are integrated with, and rely on, the other components of NLPQ, a description of those modules will be presented in this chapter. First, however, the basic concepts related to an overview of the general system NLP, the data structure utilized, and the rule language will be discussed.

A. BASIC CONCEPTS

This section is intended to provide a general outline of the basic concepts inherent in NLP and NLPQ. A detailed discussion of this material may be found in Ref. 4.

NLP is composed of a set of FORTRAN routines and a rule language. The main program serves as a monitor and performs certain input/output operations. Subroutines compile statements in the rule language and interpret operations given by the rule statements. An entity-attribute-value data structure is utilized to hold information. The generality and usefulness of this type of data structure has been widely recognized in the fields of simulation programming systems and artificial intelligence.

The entity-attribute-value data structure is well suited for holding several types of information. For example, in the current queuing problem application, information about the various words and concepts related to queuing problems must be maintained. Relations between

words and concepts can be considered to be held in long term memory, since information of this type is necessary to carry on a dialogue about any problem.

Other information about a specific problem being described must be retained for the duration of the problem solving session. Information of this type is obtained through discourse. The problem description input by the user is first processed by the "decoding" rules. These rules serve to convert the information contained in the description to an equivalent internal representation. This type of storage can be considered as short term memory since the system need only retain it for the duration of the specific problem solving session.

Another type of information storage can be considered to be temporary or scratchpad memory. For instance, information about parts of a sentence can be discarded after the sentence has been completely processed. An important aspect of NLP is that all of these types of information are maintained in exactly the same form, i. e., entity-attribute-value. Thus, all types of information can be manipulated in the same fashion.

Rules in the rule language of NLP consist of two parts separated by an arrow. The left part generally specifies conditions which must be satisfied before the rule can be applied. The right part of the rule specifies the actions to be taken when the rule is executed. Various basic elements, known as records, establish the state of the system.

These records carry the types of information mentioned in the preceding paragraphs.

Those records which are used in a scratchpad fashion to create or modify other records are called segment records. The state conditions contained in this type of record are the most frequently tested by the rules. The rules for "decoding" specify the manner in which records are to be created from input character strings. The "encoding" rules describe the inverse conversion from records to character strings. Record-to-record transformations can be accomplished by either type of rule. Thus, the basic system deals with the conversion of information. Information in the form of a natural language character string may be transformed to an internal format, manipulated, and possibly returned to a character string. The modular sets of rules for performing these functions for NLPQ will be considered below. Since the internal problem description (IPD) is the structure to be built by the decoding rules, it will be discussed first.

B. THE INTERNAL PROBLEM DESCRIPTION

Utilizing the entity-attribute-value data structure previously discussed, the logical structure of the IPD is a set of records which contain information about the current problem being processed by NLPQ. These records represent entities such as physical objects or actions occurring in the problem. These entities have attributes which in turn have values associated with them.

A typical queuing problem sequence involves mobile entities engaging in actions (abstract entities) at various stationary entities. In most cases, each of these entities has attributes of varying complexity with specified values. Each of these records in the IPD is a member of one of seven lists. Actions are members of the action list ('ACTNLIST'), mobile and stationary entities are members of the 'MOBLIST' and 'STALIST', respectively. The other lists are: the distribution list ('DSTRLIST'), the successor descriptor list ('SCSRLIST'), the miscellaneous list ('MISCLIST'), and the unit list ('UNITLIST'). Each of these lists is a "named record". Named records provide the long term memory capability previously described; that is, they contain word and concept information pertinent to the application. These particular list records, however, are used to hold information about a specific problem. As entities are encountered during decoding, records are created and linked into the appropriate lists.

The concept structure created by the named records provides the framework of words and their semantic content necessary for discourse. As such they represent the system's vocabulary and knowledge of the relationships between words and concepts. Using this general knowledge, a "mental image" (the IPD) of the specific problem entered by the user can be obtained. The IPD can be considered to be a specific problem instance in the domain of the queuing conceptual structure.

C. DECODING THE PROBLEM DESCRIPTION

In NLPQ the decoding rules specify how input text is to be converted into equivalent information in the form of records. This conversion is performed within the framework of the Stratificational Grammar theory. Within this theory several levels (strata) of language structure exist. Three levels have been utilized in the NLPQ application; the morphological, lexological, and the semological levels. The "morphology" is concerned with the manner in which characters are put together to form words. As such, it is highly dependent on the particular language being used. The "lexology" deals with the way in which words are put together to form phrases, clauses, and sentences. Different grammatical orderings required by different languages result in language dependency at this level also. The semological level, however, is concerned with relationships and meanings. Thus elements at this level are relatively language independent. The decoding process then involves applying morphological and lexological rules first in processing the input text. Semological rules are then utilized to develop the structure representing the meaning of the text, the IPD.

The general format of a decoding rule is:

SEGMENT TYPE (COND 1, 2, ...) SEGMENT TYPE (COND 1, 2, ...) ...
→ SEGMENT TYPE (ACTN 1, 2, ...)

Thus a decoding rule specifies what to do when a particular series of segment types (satisfying the given conditions) is found while processing

the input text. Rule application results in the creation of the segment type on the right and performance of those actions indicated. The conditions specified on the left side of the rule are known as "condition specifications". The actions performed in creation of the new segment on the right side are known as "creation specifications". Both types of specification elements have access to and can manipulate any information in the system.

An illustration of some of these features can be seen in the following rule from the decoding morphology:

VERBS(ING) I N G \longrightarrow VERBP(SUP(VERBS), PRESPART)

The left part of the rule is made up of four segment types, a verb stem (VERBS) segment and three "character" segments. The condition specification for the verb stem requires that the ING indicator in that segment be "on". Indicators are binary-valued and indicate the presence or absence of certain attributes. The right part of the rule defines the conversion to be made when a series of segment types satisfying the left part is encountered. In this case, a new verb part (VERBP) segment is created which is in the same superset as the verb stem and has a present participle indicator on.

Using this basic scheme, information is extracted from the input text and used to build the IPD. Once the semantic content of the user's dialogue has been transformed to the internal format, it can be manipulated in several ways.

D. ENCODING FROM THE IPD

Since encoding is basically the inverse process of decoding, encoding rules are essentially the inverse of decoding rules. Information is manipulated from the semological level, through the lexological and morphological strata to a natural language text output. A generalized format for an encoding rule is:

SEGMENT TYPE (COND 1,2,...) \longrightarrow
SEGMENT TYPE (ACTN 1,2,...) SEGMENT TYPE (ACTN 1,2,...) ...

In this case, the rule specifies the sequence of segment types (with corresponding actions) to be created when a segment type satisfying the appropriate condition specifications is encountered.

Both encoding and decoding rules have the ability to perform record-to-record information conversion, as previously stated. One modular set of encoding rules known as the MASSAGER [7] is utilized in this way to set default values in the IPD. Certain assumptions about the problem may be made by the user during the discourse. For example, if the number of units of storage capacity required by a mobile entity has not been mentioned, it is probable that an assumption of one unit has been made by the user. The purpose of the MASSAGER is to inspect the IPD after decoding is completed and set default values in those instances where non-controversial assumptions can be made. Other functions include the consolidation of redundant information and the deletion of certain attributes required only for decoding purposes.

Another encoding rule module known as the INTERROGATOR [8] serves to inspect the IPD for missing or erroneous information. Copies of the action records maintained in the IPD are accessed through the 'ACTNLIST' and tested to ensure that they have all of the attributes required for processing by the GPSS/X-VECTOR rule module. When incorrect or missing information is detected, the INTERROGATOR sets up the segment form of the question to be asked and invokes the ENGLISH encoding module to actually produce the question. Information provided by the user is then decoded and the inspection of the IPD continues. The INTERROGATOR may also be utilized to enter the problem in a question-answer fashion. When the necessary information pertaining to the problem has been established, a message is encoded indicating completion of the problem statement.

At any point during the discourse the user may request a statement of the problem as the system "sees" it. The English encoding module [6] provides the conversion necessary to produce an English description of the problem from the information contained in the IPD. The generality of this module also permits the handling of statements to the user generated by other modules such as the INTERROGATOR.

The GPSS/X-VECTOR encoding module [10] is utilized to create a GPSS program and initialize the data structure used by the simulation routine. This data structure, called the X-vector, contains the information necessary to perform the simulation. In addition, it is used throughout the simulation to maintain the required statistics in much the same manner as the internal tables of GPSS [11]

Once the problem has been completely specified, the user may request that a GPSS program be written for the current problem. The semological rules of this module examine the IPD and produce segments which roughly correspond to the statements of a GPSS program. Rules in the lexology further process these segments and their constituents into other segments in the appropriate order for a GPSS program. The morphological rules then produce the corresponding GPSS output.

Similar rules in the X-vector lexology and morphology place pertinent information into the X-vector. Figure 1 (taken from ref. 10) shows an initial X-vector produced by these rules. With the information in the X-vector the simulation can be performed.

E. THE SIMULATION ROUTINE

This FORTRAN routine, originated by Williams [9], performs a GPSS-like simulation based on the information contained in the X-vector. The initial portion of the X-vector contains "parameters" for the simulation routine. These parameters are assigned fixed locations in the vector. They consist of variables such as the random number seeds to be used, pointers to the various directories, entity allocation counts, clock time, etc. The latter portion of the vector contains allocated space for the various GPSS entities (i.e., STORAGES, QUEUES, TABLES, FUNCTIONS, VARIABLES, SAVEVALUES, and BLOCKS). The directories associated with these entities are also allocated space in this section of the vector. The location of these allocated

1	MODE	1	29	LAST TRANS. FEC. PTR.	0	282	VARIABLE ALLOCATION
2	TERMINATION COUNT	1	30	ERROR CODE	0	288	BLOCK ALLOCATIONS
3	SEED1	277	31	NUMBER OF VARIABLES	1		
4	SEED2	423	32	VARIABLE DIR. PTR.	347		
5	SEED3	815	33	STORAGE ALLOCATION (STAT1)		337	STORAGE 32 DIRECTORY 50
6	SEED4	121	43	QUEUE ALLOCATION (STAT1)		339	QUEUE 42 DIRECTORY 60
7	SEED5	655	51	STORAGE ALLOCATION (PUMP2)		341	TABLE 0 DIRECTORY 68 79
8	SEED6	531	61	QUEUE ALLOCATION (PUMP2)		344	FUNCTION DIRECTORY
9	SEED7	999	69	TABLE 2 ALLOCATION		348	VARIABLE DIRECTORY
10	SEED8	813	80	TABLE 3 ALLOCATION		349	BLOCK DIRECTORY
11	CLOCK TIME	0	91	EXPON FUNCTION ALLOCATION		363	TRANSACTION ALLOCATIONS
12	NUMBER OF STORAGES	2	169	NORMAL FUNCTION ALLOCATIONS			
13	STORAGE DIR. PTR.	336	262	ADDITIONAL FUNCTION ALLOCATIONS			
14	NUMBER OF QUEUES	2					
15	QUEUE DIR. PTR.	338					
16	NUMBER OF FUNCTIONS	4					
17	FUNCTION DIR. PTR.	343					
18	NUMBER OF BLOCKS	14					
19	BLOCK DIR. PTR.	348					
20	NUMBER OF TABLES	3					
21	TABLE DIR. PTR.	340					
22	NUMBER OF SAVEVALUES	0					
23	SAVEVALUE DIR. PTR.	0					
24	NUMBER OF PARAMETERS	3					
25	TRANSACTION POINTER	362					
26	FIRST TRANS. CEC. PTR.	0					
27	LAST TRANS. CEC. PTR.	0					
28	FIRST TRANS. FEC. PTR.	0					

Figure 1: Internal Structure of the X-Vector

areas is flexible and is determined by the requirements of the specific problem.

The procedure used to execute the simulation is essentially the same as that of GPSS. Raw results of the simulation, such as the cumulative time integrals for storages and queues, are stored in the X-vector elements associated with these entities. Several simulation modes comparable to the GPSS control cards SIMULATE/START, RESET, CLEAR, and START perform similar functions in the simulation routine.

III. A SAMPLE SIMULATION PROBLEM

This chapter is intended to demonstrate the capabilities of this initial version of the interactive simulation system being developed. It illustrates how a user may input a queuing problem in English, have the system question him for needed information, obtain a restatement of the problem from the computer, and have the system perform the simulation and output the requested results.

The sample given here is taken from an actual terminal session. In line with the purpose of demonstrating the system's capabilities, a wide variety of statements and questions are given. This results in an unusual amount of redundancy.

Throughout this chapter, all inputs by the user are shown in lower case, and all computer responses are shown in UPPER CASE. This is the way a sample session under CP/CMS would actually appear. Comments are interspersed where needed and are enclosed in parentheses. The sample session begins with "decode:" below.

decode:

(This is a command to NLP necessary to invoke the decoding rules for processing the input text.)

vehicles arrive at a station .

the station has just one pump.

a vehicle will leave the station immediately after arriving if the length of the line at the pump is not less than two.

otherwise, it is serviced there; then it leaves.

service times are exponential, with a mean of 5 minutes for cars and 9 minutes for trucks.

three quarters of the vehicles are cars and one fourth of them are trucks.

is the statement of the problem complete?

(This question results in inspection of the IPD for missing or erroneous information.)

HOW OFTEN DO THE VEHICLES ARRIVE AT THE STATION?

(The question-answer mode is initiated to query the user for necessary information.)

arrivals are normally distributed with a mean of 8 minutes and a standard deviation of 2 minutes.

HOW LONG SHALL THE SIMULATION BE RUN?

the basic time unit to be used in the model is 30 seconds.

(Even though this response does not answer the question, this information is still accepted into the IPD. The system will query the user again for the answer to its question.)

HOW LONG SHALL THE SIMULATION BE RUN?

eight hours.

THE PROBLEM STATEMENT IS COMPLETE.

describe the problem in english.

(This is requesting a restatement of the problem as it now appears in the IPD.)

THE VEHICLES ARRIVE AT THE STATION. THE TIME BETWEEN ARRIVALS OF THE VEHICLES AT THE STATION IS NORMALLY DISTRIBUTED, WITH A MEAN OF 8 MINUTES AND A STANDARD DEVIATION OF 2 MINUTES. 75 PERCENT OF THE VEHICLES ARE CARS, AND THE REST ARE TRUCKS. AFTER ARRIVING AT THE STATION, IF THE LENGTH OF THE LINE AT THE PUMP IN THE STATION IS LESS THAN 2, THE VEHICLES WILL BE SERVICED AT THE PUMP IN THE STATION. OTHERWISE, THE VEHICLES WILL LEAVE THE STATION. THE TIME FOR THE VEHICLES TO BE SERVICED AT THE PUMP IN THE STATION IS EXPONENTIALLY DISTRIBUTED, WITH A MEAN OF 5 MINUTES FOR THE CARS, AND 9 MINUTES FOR THE TRUCKS. AFTER BEING SERVICED AT THE PUMP IN THE STATION, THE VEHICLES LEAVE THE STATION.

THE SIMULATION IS TO BE RUN FOR 8 HOURS, USING A BASIC TIME UNIT OF 30 SECONDS.

write a gpss program for this problem.

(Production of the GPSS program also results in X-vector initialization in preparation for running the simulation. The program produced is shown on the following page.)


```

SIMULATE
RMULT      277,423,715,121,655,531,999,813
STAT1 EQU  1,F,Q
PUMP2 EQU  2,F,Q
CAR2 EQU   2,T
2 TABLE  M1,1,1,2
TRUC3 EQU  3,T
3 TABLE  M1,1,1,2
1 FUNCTION RN1,C24
0.0,0.0/0.100,0.104/0.200,0.222/0.300,0.355/
0.400,0.509/0.500,0.690/0.600,0.915/0.700,1.200/
0.750,1.390/0.800,1.600/0.840,1.830/0.880,2.120/
0.900,2.300/0.920,2.520/0.940,2.810/0.950,2.990/
0.960,3.200/0.970,3.500/0.980,3.900/0.990,4.600/
0.995,5.300/0.998,6.200/0.999,7.000/1.000,8.000/
2 FUNCTION RN2,C29
0.0,-3.000/0.012,-2.250/0.027,-1.930/0.043,-1.720/
0.062,-1.540/0.084,-1.380/0.104,-1.260/0.131,-1.120/
0.159,-1.000/0.187,-0.890/0.230,-0.740/0.267,-0.620/
0.334,-0.430/0.432,-0.170/0.500,0.0/0.568,0.170/
0.666,0.430/0.732,0.620/0.770,0.740/0.813,0.890/
0.841,1.000/0.869,1.120/0.896,1.260/0.916,1.380/
0.938,1.540/0.957,1.720/0.973,1.930/0.988,2.250/
1.000,3.000/
3 FUNCTION! P1,D2
CAR2,10/TRUC3,18/
4 FUNCTION! RN3,D2
0.750,CAR2/1.000,TRUC3/
1 FVARIABLE 16+4*FN2
*
* THE VEHICLES ARRIVE AT THE STATION.
GENERATE V1
ASSIGN 1,FN4
TEST L Q$PUMP2,2,ACT2
TRANSFER ,ACT3
*
* THE VEHICLES LEAVE THE STATION.
ACT2 TABULATE P1
TERMINATE
*
* THE VEHICLES ARE SERVICED AT THE PUMP.
ACT3 QUEUE PUMP2
SEIZE PUMP2
DEPART PUMP2
ADVANCE FN3,FN1
RELEASE PUMP2
TRANSFER ,ACT2
*
* TIMING LOOP
GENERATE 960
TERMINATE 1
START 1
END

```


perform the simulation.

SIMULATION TIME IS 960(RELATIVE), 960(ABSOLUTE).

(This message signals completion of the simulation. The user may now ask for statistical printouts or for specific information concerning the outcome.)

print the gpss statistics.

STORAGE	CAPACITY	AVERAGE CONTENTS	AVERAGE UTILIZATION	ENTRIES	AVERAGE TIME/TRAN	CURRENT CONTENTS	MAXIMUM CONTENTS
1	1	0.0	0.0	0	0.0	0	0
2	1	0.659	0.659	59	10.729	0	1

QUEUE	MAXIMUM CONTENTS	AVERAGE CONTENTS	TOTAL ENTRIES	ZERO ENTRIES	PERCENT ZEROS	AVERAGE TIME/TRANS	\$AVERAGE TIME/TRANS	CURRENT CONTENTS
1	0	0.0	0	0	0.0	0.0	0.0	0
2	2	0.278	59	38	64.407	4.525	12.714	0

\$AVERAGE TIME/TRANS = AVERAGE TIME/TRANS EXCLUDING ZERO ENTRIES

TABLE 2

ENTRIES IN TABLE		MEAN ARGUMENT	STANDARD DEVIATION		SUM OF ARGUMENTS	
43		13.628	13.175		586.000	
UPPER LIMIT	OBSERVED FREQUENCY	PER CENT OF TOTAL	CUMULATIVE PERCENTAGE	CUMULATIVE REMAINDER	MULTIPLE OF MEAN	DEVIATION FROM MEAN
1	3	6.98	6.98	93.02	0.073	-0.958
OVERFLOW	40	93.02	100.00	0.00		
AVERAGE VALUE OF OVERFLOW		14.650				

TABLE 3

ENTRIES IN TABLE		MEAN ARGUMENT	STANDARD DEVIATION		SUM OF ARGUMENTS	
18		17.444	12.641		314.000	
UPPER LIMIT	OBSERVED FREQUENCY	PER CENT OF TOTAL	CUMULATIVE PERCENTAGE	CUMULATIVE REMAINDER	MULTIPLE OF MEAN	DEVIATION FROM MEAN
1	2	11.11	11.11	88.89	0.057	-1.301
OVERFLOW	16	88.89	100.00	0.00		
AVERAGE VALUE OF OVERFLOW		19.625				

CURRENT EVENTS CHAIN			NBA	MARK-TIME	P1	P2	P3	P4	SI	TI
TRANS	BDT	BLOCK								

FUTURE EVENTS CHAIN			NBA	MARK-TIME	P1	P2	P3	P4	SI	TI
TRANS	BDT	BLOCK								
3	964	1	2	964	0	0	0	23658912	T	T
2	1920	13	14	1920	0	0	0	372	T	T

print the storage statistics.

STORAGE	CAPACITY	AVERAGE CONTENTS	AVERAGE UTILIZATION	ENTRIES	AVERAGE TIME/TRAN	CURRENT CONTENTS	MAXIMUM CONTENTS
1	1	0.0	0.0	0	0.0	0	0
2	1	0.659	0.659	59	10.729	0	1

print the pump statistics.

STORAGE	CAPACITY	AVERAGE CONTENTS	AVERAGE UTILIZATION	ENTRIES	AVERAGE TIME/TRAN	CURRENT CONTENTS	MAXIMUM CONTENTS
2	1	0.659	0.659	59	10.729	0	1

QUEUE	MAXIMUM CONTENTS	AVERAGE CONTENTS	TOTAL ENTRIES	ZERO ENTRIES	PERCENT ZEROS	AVERAGE TIME/TRANS	SAVERAGE TIME/TRANS	CURRENT CONTENTS
2	2	0.278	59	38	64.407	4.525	12.714	0

SAVERAGE TIME/TRANS = AVERAGE TIME/TRANS EXCLUDING ZERO ENTRIES

print the pump queue statistics.

QUEUE	MAXIMUM CONTENTS	AVERAGE CONTENTS	TOTAL ENTRIES	ZERO ENTRIES	PERCENT ZEROS	AVERAGE TIME/TRANS	SAVERAGE TIME/TRANS	CURRENT CONTENTS
2	2	0.278	59	38	64.407	4.525	12.714	0

SAVERAGE TIME/TRANS = AVERAGE TIME/TRANS EXCLUDING ZERO ENTRIES

print the truck statistics.

TABLE 3

ENTRIES IN TABLE		MEAN ARGUMENT		STANDARD DEVIATION		SUM OF ARGUMENTS	
18		17.444		12.641		314.000	
UPPER LIMIT	OBSERVED FREQUENCY	PER CENT OF TOTAL	CUMULATIVE PERCENTAGE	CUMULATIVE REMAINDER	MULTIPLE OF MEAN	DEVIATION FROM MEAN	
1	2	11.11	11.11	88.89	0.057	-1.301	
OVERFLOW	16	88.89	100.00	0.00			
AVERAGE VALUE OF OVERFLOW		19.625					

print the car table statistics.

TABLE 2

ENTRIES IN TABLE		MEAN ARGUMENT		STANDARD DEVIATION		SUM OF ARGUMENTS	
43		13.628		13.175		586.000	
UPPER LIMIT	OBSERVED FREQUENCY	PER CENT OF TOTAL	CUMULATIVE PERCENTAGE	CUMULATIVE REMAINDER	MULTIPLE OF MEAN	DEVIATION FROM MEAN	
1	3	6.98	6.98	93.02	0.073	-0.958	
OVERFLOW	40	93.02	100.00	0.00			
AVERAGE VALUE OF OVERFLOW		14.650					

what is the current contents of the pump?

THE CURRENT CONTENTS OF THE PUMP IN THE STATION IS 0.

what is the s of the pump?

THE CURRENT CONTENTS OF THE PUMP IN THE STATION IS 0.

(Note that the above questions to the system are equivalent. Questions can be asked by specifying the GPSS standard numerical attribute ("s") or by specifying that attribute in English ("current contents").)

what is the sm of the pump?

THE MAXIMUM CONTENTS OF THE PUMP IN THE STATION IS 1.

what is the sa of the pump?

THE AVERAGE CONTENTS OF THE PUMP IN THE STATION IS 0.659.

what is the st at the pump?

THE AVERAGE SERVICE TIME AT THE PUMP IN THE STATION IS 10.729.

what is the qz at the pump?

THE NUMBER OF ZERO ENTRIES AT THE PUMP IN THE STATION IS 38.

what is the qx at the pump?

THE AVERAGE WAITING TIME EXCLUDING ZERO ENTRIES AT THE PUMP IN THE STATION IS 12.714.

what is the maximum line at the pump?

THE MAXIMUM LINE LENGTH AT THE PUMP IN THE STATION IS 2.

what is the current line at the pump?

THE CURRENT LINE LENGTH AT THE PUMP IN THE STATION IS 0.

what is the mean transit time of the cars?

THE MEAN TRANSIT TIME OF THE CARS IS 13.627.

what is the average line at the pump?

THE AVERAGE LINE LENGTH AT THE PUMP IN THE STATION IS 0.278.

what is the average utilization of the pump?

THE AVERAGE UTILIZATION OF THE PUMP IN THE STATION IS 0.659.

what is the average waiting time at the pump?

THE AVERAGE WAITING TIME AT THE PUMP IN THE STATION IS 4.525.

what is the average time at the pump?

THE AVERAGE SERVICE TIME AT THE PUMP IN THE STATION IS 10.729.

continue the simulation.

(This is comparable to the encountering of a second or subsequent "start" card in GPSS. The simulation progresses from the point at which it previously terminated.)

SIMULATION TIME IS 1920(RELATIVE), 1920(ABSOLUTE).

what is the average utilization of the pump?

THE AVERAGE UTILIZATION OF THE PUMP IN THE STATION IS 0.654.

what is the average waiting time at the pump?

THE AVERAGE WAITING TIME AT THE PUMP IN THE STATION IS 6.111.

what is the average time at the pump?

THE AVERAGE SERVICE TIME AT THE PUMP IN THE STATION IS 10.930.

(The three questions above were asked in order to show the effect the longer simulation time had on the previous results.)

continue the simulation

SIMULATION TIME IS 2880(RELATIVE), 2880(ABSOLUTE).

what is the average utilization of the pump?

THE AVERAGE UTILIZATION OF THE PUMP IN THE STATION IS 0.701.

what is the average waiting time at the pump?

THE AVERAGE WAITING TIME AT THE PUMP IN THE STATION IS 8.674.

what is the average time at the pump?

THE AVERAGE SERVICE TIME AT THE PUMP IN THE STATION IS 12.242.

reset and start the simulation.

(This is similar to using the control cards "reset/start" in GPSS.)

SIMULATION TIME IS 960(RELATIVE), 2840(ABSOLUTE).

what is the average utilization of the pump?

THE AVERAGE UTILIZATION OF THE PUMP IN THE STATION IS 0.768.

what is the average waiting time at the pump?

THE AVERAGE WAITING TIME AT THE PUMP IN THE STATION IS 9.732.

what is the average time at the pump?

THE AVERAGE SERVICE TIME AT THE PUMP IN THE STATION IS
12.982.

clear and restart the simulation.

(This is similar to using the control cards "clear/start"
in GPSS.)

SIMULATION TIME IS 960(RELATIVE), 960(ABSOLUTE).

what is the average utilization of the pump?

THE AVERAGE UTILIZATION OF THE PUMP IN THE STATION IS
0.732.

what is the average waiting time at the pump?

THE AVERAGE WAITING TIME AT THE PUMP IN THE STATION IS
14.132.

what is the average time at the pump?

THE AVERAGE SERVICE TIME AT THE PUMP IN THE STATION IS
13.557.

are there two pumps in the station?

(This question results in an inspection of the IPD to
determine the number of pumps.)

NO, THERE IS 1 PUMP IN THE STATION.

the station should have two pumps.

(This statement results in an updating of the IPD to
reflect this new information.)

how often does a vehicle arrive?

THE TIME BETWEEN ARRIVALS OF THE VEHICLES AT THE STATION IS NORMALLY DISTRIBUTED WITH A MEAN OF 8 MINUTES AND A STANDARD DEVIATION OF 2 MINUTES.

the mean of the time between arrivals should be 3 minutes,
and the deviation of the time between arrivals should be 1
minute.

develop the x vector.

(Since the IPD has been altered, the X-vector must be reinitialized prior to simulation to reflect the changes made. In this case a GPSS program printout is not desired.)

perform the simulation.

(This results in the simulation being performed with the X-vector for the modified problem.)

SIMULATION TIME IS 960(RELATIVE), 960(ABSOLUTE).

what is the average utilization of the pumps?

THE AVERAGE UTILIZATION OF THE PUMPS IN THE STATION IS 0.093.

what is the average waiting time at the pumps?

THE AVERAGE WAITING TIME AT THE PUMPS IN THE STATION IS 3.882.

what is the average time at the pumps?

THE AVERAGE SERVICE TIME AT THE PUMPS IN THE STATION IS 10.529.

IV. IMPLEMENTATION

Achieving the objectives of this research required making additions and changes to the FORTRAN routines of NLPQ and to the rules and declarations processed by those routines. This chapter, which is intended to explain the modifications, has been divided into two sections. The first section describes the FORTRAN modifications, and the second section describes the rule modifications.

A. FORTRAN ROUTINES AND MODIFICATIONS

The main FORTRAN programming effort involved major alterations and additions to the simulation routine. A few additional modifications to NLP were needed, however, in subroutines PRINT, ENCODE, and CRSEG. A discussion on each of these four areas follows.

1. NLP Modifications to Allow X-Vector Read and Write

This section was added simply for the convenience of the user. It gives the user a method for saving the contents of the X-vector as a binary file for use in a later terminal session. By doing this the user does not have to duplicate his efforts from a previous session to arrive at the same point in a given simulation. He need only initialize the current X-vector with the contents of the previous X-vector.

The FORTRAN routine for performing this function is shown in Appendix A. This routine was added as entry point XRDWR

("X-vector Read/Write") in subroutine PRINT. The routine may be invoked at any time as a command to NLPQ. The format for a call to XRDWR from the terminal is

X ^ READ ^ f: or X ^ WRITE ^ f: ,

where "f" is an integer number of any available file under CP/CMS and "^" denotes at least one blank space. As an example, the command "X WRITE 3:" would cause the current contents of the X-vector to be written into file FT03F001 as a binary file.

2. Establishing a Linkage for Communication

The rule language of NLP utilizes declared ROUTINES to communicate with the various FORTRAN subroutines. The appearance of a routine name in a rule causes execution of the code for that particular routine in subroutine CRSEG ("Create Segment"). To establish communication between the rules and the simulation subroutine, therefore, it was necessary to add an additional routine in subroutine CRSEG to communicate with each entry point in the simulation subroutine. The FORTRAN code for establishing this communication is given in Appendix A.

The four entries into the simulation subroutine (SIMULT, SIMOUT, SETIND, and SPSTAT) will be discussed in detail later in this section. At this point it is sufficient to say that SIMULT ("Simulation") and SIMOUT ("Simulation Output") require no information from the rule segment being processed. In addition, they are

called merely to perform their functions and not to return a value. Hence, execution of these routines in CRSEG simply results in a call to the appropriate entry point in the simulation subroutine.

Both SETIND ("Set Indicators") and SPSTAT ("Specific Statistics"), however, require the value of certain attributes of the segment being processed to be passed as arguments. In addition, SPSTAT returns a value which must be made available to the segment being processed. The additional coding in CRSEG for these two routines sets up this communication.

3. Modified Output Routine

The output routine in subroutine ENCODE was originally implemented to handle only integer half-word values (or integer half-word values expressed in parts per thousand) and, therefore, could output only integers in the range -32768 to 32767 or real values in the range -32.768 to 32.767. This was acceptable when the only output from the system was a GPSS program. With the incorporation of the simulation routine and the ability to actually run the simulation at the terminal and question the system for such items as the mean transit time or the average utilization, however, this limitation became unacceptable. As a result, the output routine was rewritten to handle the magnitude of any integer or real value which could be stored in a full-word on the IBM 360. The modified section of ENCODE pertaining to the output of numerical values is shown in Appendix A.

To output a numerical value from an OUTPUT segment, the segment must have an attribute (ATTR or @) 14, 15, or 16. An integer or real value will then be output in accordance with one of the four cases described below.

Case 1. If an @14 cell is present and the TYPE of the cell is "0", then the value in the address (ADDR) field is taken to be a half-word integer.

Case 2. If an @14 cell is present and the TYPE of the cell is "1", then the value of the ADDR field is taken to be a pointer to another cell whose value is a full-word found in the ADDR and LINK fields. This value is taken to be an integer unless a "1" is present in the ATTR field of the same cell, in which case the number is considered to be floating point.

Case 3. If an @15 cell is present, then the value in the ADDR field is taken to be a real number expressed as a half-word integer in parts per thousand.

Case 4. If an @16 cell is present, then the value in the ADDR field is taken to be a pointer to another cell whose value is a full-word found in the ADDR and LINK fields. This number is evaluated as in Case 2 above.

These four methods of handling numerical values in an OUTPUT segment are illustrated in Figure 2.

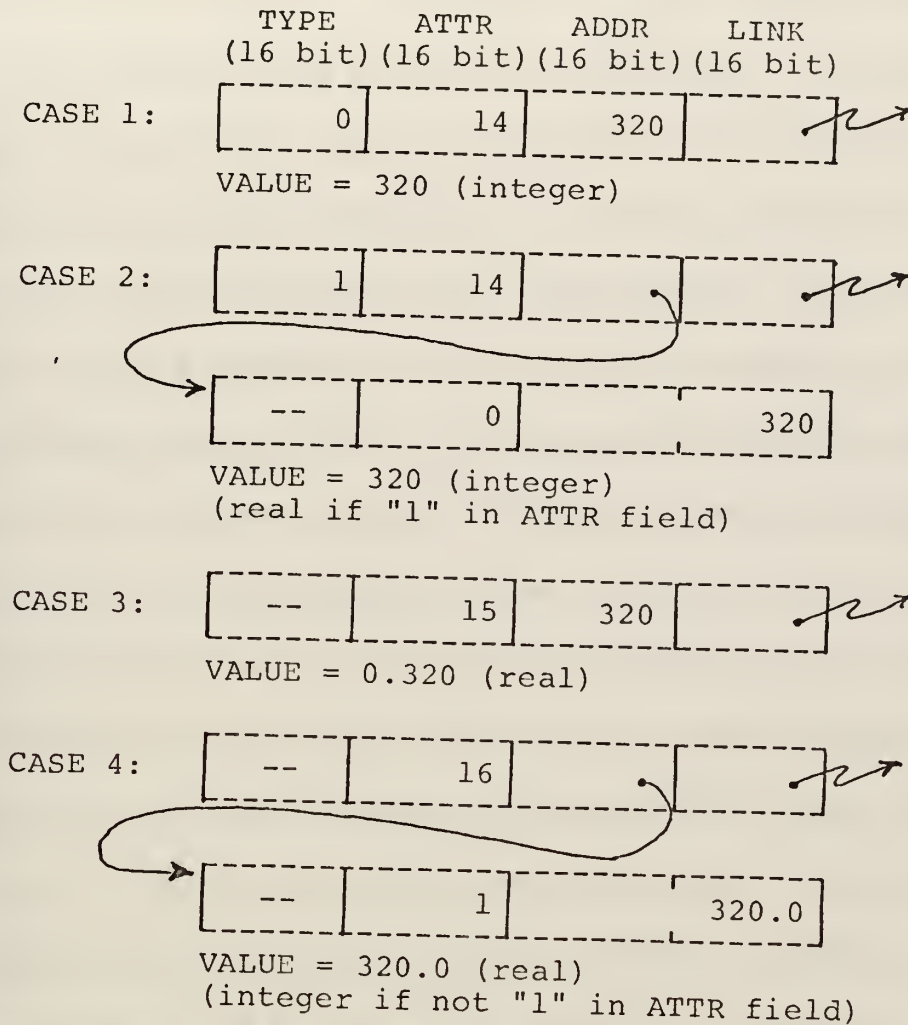


Figure 2: Numerical Evaluation for OUTPUT Segment.

4. The Simulation Subroutine

As previously mentioned, modifications to the simulation subroutine (called SIMULT) constituted the major FORTRAN programming effort involved in the preparation of this thesis. A complete listing of the subroutine is given in Appendix C. This listing is preceded by an extensive dictionary of variables (Appendix B) to assist the reader in understanding the logic of the program. Even though SIMULT is only a subroutine of NLPQ it is quite extensive and requires a considerable amount of core. The source deck contains approximately 1600 cards, and a region of 250K is needed for compilation under OS/360. A slightly larger region is required under CP/CMS. Approximately one and one-half minutes of CPU time are required for compilation using the FORTRAN G-level compiler, and the resultant object module occupies approximately 40K bytes in the IBM 360. The subroutine contains four entry points. It can be accessed by calls to SIMULT, SIMOUT, SETIND, or SPSTAT. Each of these sections will be discussed individually at this point.

a. SIMULT

This is the main entry into the simulation subroutine. A call to SIMULT is a request to perform the simulation based on the information found in the X-vector. For this reason the user must ensure that the vector has been properly initialized prior to requesting that a simulation be performed. If the user desires to continue the problem from a previous session in which the contents of the

vector were saved, he may utilize the X READ feature described previously to initialize the current X-vector. If this is not the case, however, the user must either tell the system to develop an X-vector, or he must tell the system to write a GPSS program, in a manner to be described later. The former case will result in only the initialization of the X-vector based on the information contained in the IPD. No output will be sent to the terminal. The latter case will result in both the GPSS program being output and the concurrent initialization of the X-vector. It is important to note that even though NLP gives the user the capability of saving the internal problem description and reinitializing the system with that IPD at a later date, this procedure does not reinitialize the X-vector.

The SIMULT section is basically the simulation routine written by R. J. Williams [9]. The basic logic flow and the structure of the various statistical entity layouts described by Williams were retained in the implementation of the simulation capability to NLPQ. The reader interested in following the logical flow of the SIMULT listing is advised, therefore, to reference Williams' work for the physical layout of the various statistical entities (STORAGE, QUEUE, TABLE, etc.).

Major alterations to the basic flow of transactions through the system were made upon incorporation of the simulation routine into NLPQ. For example, a reordering of the sequences involved for merging transactions of equal priority into their

appropriate positions on the current and future events chains was needed in the verification phase to ensure that transactions would be executed in the same sequence as is done in GPSS. Modifications were also required in the procedures associated with determining when a status change has occurred within the system. These modifications further altered the original flow of transactions since these procedures determine at what points in the simulation a scan of the current events chain should be continued from its previous position and at what points it should be restarted from the beginning of the chain.

Additional modifications included the addition of the entire section pertaining to the updating of the system performed during the final time interval prior to terminating the simulation. This area had been completely omitted in Williams' work but was needed to ensure agreement between the statistical outputs of GPSS and subroutine SIMULT. This section primarily performs an update of the STORAGE and QUEUE statistics to reflect the effect of the simulation time involved between the time the storage or queue last changed status and the time the simulation was actually terminated. The ability to output a selected portion of the X-vector was also included in this section.

Further modification to the original simulation subroutine involved the addition of code throughout the routine to avoid system interrupts, such as divide checks, when working with empty

storages, queues, and tables. Similar modifications were also required throughout the routine to ensure that null pointers in the various directories were recognized as such and not used as valid indices when storing or retrieving information pertaining to the subscripted X-vector.

The ability to properly handle clear, reset, and continue commands by the user required some alterations to properly re-initialize the allocated storages, queues, and tables. In addition the procedure for processing the TERMINATE block was modified to ensure replacement of the timing loop generate block on the future events chain. The use of both an absolute and relative clock during the simulation was deleted and replaced by a single clock (absolute) for use during the simulation. A base clock is set in the RESET area to allow computation of the relative time in the two instances in which a relative time is required, i. e., (1) the "C1" Standard Numerical Attribute and (2) the message signaling completion of the simulation.

The section for argument evaluation was altered and expanded somewhat to allow requests for specific statistics (entering the routine via SPSTAT) to access that area of code for evaluation of the statistical information requested. Most of the remaining alterations to the original routine were either minor in nature or made simply in the interest of cleaner coding.

SIMULT is entered upon a user request to perform the simulation or a request to clear, reset, or continue the simulation. Initialization of the simulation model is then performed, if necessary, based on the type of request made. This initialization is analogous to that performed by GPSS upon encountering the control cards SIMULATE/START, CLEAR, RESET, and START. The algorithms which direct the flow of transactions through the system from this point are essentially the same algorithms used in GPSS. Since the results produced by the program, as well as the information needed to perform the simulation, are contained in the X-vector, execution of SIMULT results in an X-vector altered to reflect the current status of the simulation. The results, therefore, are readily available to be accessed in the event the user requests information concerning the outcome of the simulation. Assuming no error conditions are encountered during the simulation, the only output from a SIMULT entry will be a message giving the absolute and relative simulation clock times. This message signals completion of the simulation.

b. SIMOUT

The "Simulation Output" routine is invoked whenever the user requests GPSS-like statistical information. The format of the information output by SIMOUT is practically identical to that of GPSS. Unlike GPSS, however, SIMOUT has the capability of providing the user with (1) an entire statistical printout (including storage and queue statistics, tables, savevalues, the current events chain, and the future

events chain), (2) a single block of any of the statistics just mentioned (the storage statistics alone, for example), (3) a single line of storage or queue statistics for a single table of several tables (for example, the queue statistics for a specified stationary entity), or (4) the queue and storage statistics for any specified stationary entity (such as a pump). For example, a user request to "print the GPSS statistics" will result in (1) above. Similarly, "print the storage statistics" will result in (2) and "print the pump queue statistics" will result in (3). If the user's request is "print the pump statistics", the SIMOUT routine will output both the storage and queue statistics for the pump. This will be explained more fully later.

The SIMOUT routine has no access to the current segment being processed. Yet the routine needs to know exactly which lines are to be output and which are not. This information is obtained from a vector called STATSW ("Statistic Switches"). This vector is local to the SIMULT routine and, hence, can be accessed by both SIMOUT and SETIND. It is the function of SETIND (which will be described later in this section) to set the proper statistic switches for SIMOUT. A call to SIMOUT, therefore, must always be preceded by a call to SETIND. These calls, however, result from the processing of the input text and are made from subroutine CRSEG. They are completely transparent to the user.

STATSW (shown in figure 3) is a four element, real*8 vector. The first three elements contain indicators for storages, queues, and tables, respectively. The rightmost 55 bits in each

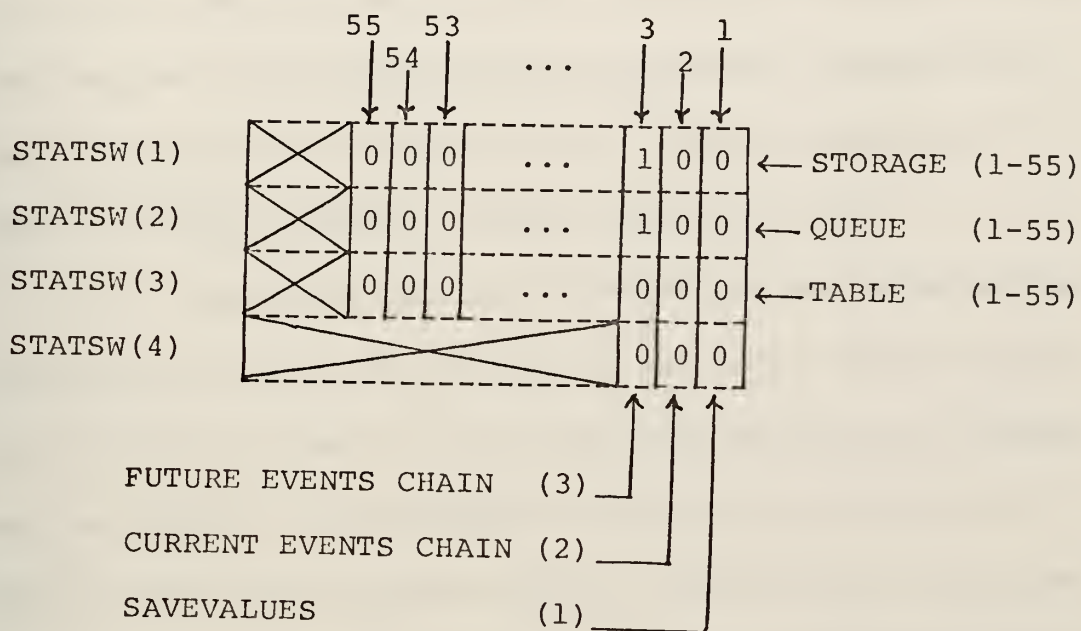


Figure 3: Statistic Switch Vector.

element are used to indicate which storage, queue, or table should be output. If the first element in STATSW has a "1" in bit position two and zeros elsewhere, for example, the SIMOUT routine would output the storage statistics for the stationary entity which has an identification number (IDNO) of two in the IPD. The storage statistics for the remaining stationary entities would not be printed. Indicators for queues and tables are treated similarly, with the exception that the IDNO for the table refers to a mobile entity in the IPD.

The fourth element in STATSW contains indicators for savevalues and the current and future event chains. Only the right-most three bits are used. A "1" in the third bit position of element four, for example, would indicate the future events chain is to be output. The bit pattern shown in figure 3 indicates that storage and queue statistics for the stationary entity having an IDNO equal to 3 is to be output. No other statistics are to be printed.

Upon entry into SIMOUT a call is immediately made to subroutine GBITS ("Get Bits"). This routine returns the value of bits 1 through 55 of the first STATSW element. A zero value indicates no storage statistics are to be output. In this case a branch is made around the "output storage" area to the "output queue" area. If the value returned is not zero, however, then one or more lines of storage statistics are to be output. SIMOUT, therefore, outputs the storage headings and then begins a search to determine which of the storages are to be output. This is done by successive calls to GBITS to obtain

the value of bit positions 1, 2, 3, ... (up to the number of storages being maintained in the system). If the individual bit value is "0", that storage is bypassed. If the bit value is "1", however, SIMOUT utilizes the current information in the X-vector for that particular storage to calculate and output the statistics required. When this has been done for each storage, the program falls through to the "output queue" area.

The procedure for determining which queues and tables are to be output, if any, is the same as that for storages. Upon falling through to the "output savevalues" area, however, only one call to GBITS is made to determine if bit position one of STATSW(4) is on. If this is the case, all of the savevalues are listed with their respective values.

The procedure for determining whether the current and future events chains (bit positions two and three) are to be output is essentially the same as that for savevalues. The noticeable difference in output of the chains is that both are handled in the same area in order to avoid duplication of code.

Upon completion of the statistical outputs, SIMOUT zeros all four elements of STATSW prior to returning to CRSEG. This is required to prevent unwanted statistics from being printed if the user later requests further information requiring another call to SIMOUT.

c. SETIND

As mentioned previously, it is the function of SETIND ("Set Indicators") to set the desired bits in the STATSW vector to enable SIMOUT to output the proper statistics. It is a function of the decoding rules to determine the content of the English request and issue a call to SETIND, telling the routine which bits are to be set. Like all calls to the simulation routine, this call is made via sub-routine CRSEG.

SETIND must receive two parameters from the calling segment. These parameters are (1) the row in STATSW which is affected and (2) the bit position (or positions) within that row which is to be set. Since a single request for statistics by the user may involve the setting of a single row or multiple rows and may involve the setting of a single bit or all the bits within a given row, this capability has been included in SETIND in order that these multiple settings might be performed by a single call to SETIND. This is accomplished by the manner in which SETIND handles the calling arguments. If the requested row is in the range 1 through 4, the routine assumes the row specified is to be set. If the calling argument for the row is 5, however, the routine assumes that the bit (or bits) specified are to be set in both rows 1 and 2 of STATSW. This condition is common to a request for statistics of stationary entities (which have both storage and queue statistics). A calling argument of 6 specifies that all the bits of each element are to be set. This corresponds to a total

GPSS-like printout. If the second calling argument (the bit position to be set) is in the range 1 through 55, that single bit position is set. If the second argument is greater than 55, however, then all of the bits of the specified row are turned on. This condition is common when issuing a request for statistics without specifying an associated stationary or mobile entity.

The actual setting of the bit positions is not necessarily performed by SETIND. SETIND merely analyzes the request to determine which bits are to be set. If an entire row in STATSW is to be turned on, SETIND will perform the function. If only a single bit is to be turned on, however, SETIND issues a call to subroutine PBITS ("Put Bits") to set the desired bit position. Once the desired bits have been set, SETIND returns to the calling routine, CRSEG.

d. SPSTAT

A user request for a single item of statistical information is processed by the rules in a manner similar to any other question to the system. The value of most items of statistical interest, however, is not available in the IPD. When it is determined in the processing of the text that one of these values is being requested, attributes are set in the segment being processed to designate the type of value being requested and the IDNO of the entity for which the value is to be computed. A call is then made through CRSEG to SPSTAT ("Specific Statistics") passing these attributes as parameters.

SPSTAT sets an initial default value of zero to be returned in the event an error condition occurs (such as a request for statistical information prior to performing the simulation). The routine then sets an entry point flag and branches into the SIMULT argument evaluation section to compute the desired statistic. As previously mentioned, this section in SIMULT was modified to be able to process inputs from both entry points. A SIMULT entry into this section causes all real argument values to be truncated to integer values. An SPSTAT entry, however, requires that real values be retained and returned as floating point.

Upon completion of argument evaluation, the entry point flag directs the logical flow back to SPSTAT. The bit pattern of the value is set into an integer word and a flag is set to specify whether the value being returned should be interpreted as an integer or a decimal result. The requested value and the flag are then passed back to CRSEG which inserts the information into the current segment record to be output later in the encoded text.

B. RULE ADDITIONS AND MODIFICATIONS

Integration of the simulation routine and the ability to query the system as to the results of the simulation required several modifications and additions to the existing rule modules. Expansion of the concept structure by the addition of named record definitions was also necessary to allow questions relating to the simulation results.

The changes and additions made are shown in Appendix D and are discussed in the following sections.

1. Named Records

Several named records in the form of English words with their associated part-of-speech (PS) attributes were added to facilitate recognition of these words by the system. The GPSS entities, storage, queue, and table, were also declared and assigned numerical codes which correspond to the position of their respective indicators in the STATSW vector previously described. A superset relation is also established to identify these words as elements of the set 'GPSSSENTY' ("GPSS entity").

The remaining named record definitions identify the various GPSS "Standard Numerical Attributes" (SNA's) as members of the set 'GPSSATTR' ("GPSS attribute"). Each of these records also contains an SNACODE attribute with an associated value. This value is passed to the SPSTAT routine by the rules in those instances where a specific statistic is desired. The CHARS attribute contains the SNA name in character form to be used in encoding responses to the user's questions.

2. Simulation Control Commands

To permit interactive control of the simulation with regard to the various modes of operation, several semological decoding rules were required. These rules are basically "key word" rules which serve to test the input string for the presence of one or more

key words. Rules of this type have a left segment type of KWDSSENT. The condition specifications of these rules indicate the key words necessary for rule execution. For example, the presence of the key word "perform", "simulate", or "run", in the user's request results in execution of the simulation routine in the SIMULATE/START mode. Thus the commands; "Perform the simulation.", "Simulate the system.", or just "run.", are equivalent and result in the simulation being run.

The keywords "reset", "clear", and "continue" are handled in a similar manner. Thus by using commands such as "Reset and start the simulation.", "Clear the model.", or "Continue the simulation.", the user can control the mode of the simulation. The key word rules which handle these cases set the mode indicator of the X-vector (X(1)) to the appropriate value and reset the termination count. In the present application, the termination count is set to 1 since the GPSS/X-VECTOR rules use a "timing" transaction to terminate the simulation. The simulation is automatically restarted once these modifications have been made.

Several other functions are also performed by the key word rules. The key words "gpss" or "vector" occurring in the input command result in the initialization of the X-vector from the IPD. If only the key word "vector" is present, for example "Develop the x vector.", the GPSS program will be suppressed. If "gpss" is present, both the GPSS program and the X-vector initialization will

result. The presence of the key words "print" and "gpss" combined with the absence of the key word "program" produce a complete GPSS statistical listing. Thus, "Print the gpss statistics." and similar constructions produce output with the complete results of the simulation. The key words "print", "current", and "events" appearing in the input text result in a printout of the transactions on the current events chain. Omission of, or substitution for, the key word "current" produces a listing of the future events chain. Combination of key words which satisfy more than one rule (e. g., "Reset and run.") will result in execution of the first applicable rule based on their physical order as shown in Appendix D.

3. Producing Selective Simulation Results

Several rules were added to NLPQ to give the user the opportunity to request certain portions of the GPSS-like statistical printout. The general format for commands of this type is:

PRINT {THE} $\left[\begin{array}{l} \text{stationary entity} \\ \text{mobile entity} \end{array} \right] \left\{ \text{GPSS entity} \right\}$ STATISTICS.

Thus commands of the form "Print storage statistics." result in that portion of the statistical printout related to the GPSS entity specified; in this case, the storages. The GPSS entities which can presently be employed are "storage", "queue", and "table". The command "Print pump statistics." satisfies the general format and produces all statistical output related to the stationary entity "pump". In the present application, the statistical output produced for stationary

entities is the appropriate line of storage and queue statistics, with their respective headings. Substitution of a mobile entity in the same form (e.g., "Print truck statistics.") produces table statistics for that mobile entity.

Further selectivity can be obtained by supplying more optional information as in the command "Print the pump queue statistics." In this instance only the line of output associated with the queue at the pump will be printed. The corresponding command for mobile entities (e.g., "Print the truck table statistics.") is equivalent to the earlier mobile entity command and also results in a table. Care must be taken when utilizing this form to ensure that the selection of entities is compatible. Stationary entities must be used in context with the GPSS entity "storage" or "queue". Mobile entities require the use of the GPSS "table" entity. Incorrect sequences may result in alternate statistics or none at all.

The rules for processing these "print commands" are included in the portion of the "Lexology for Decoding English" shown in Appendix D. The processing is based on the appearance of noun phrases which are elements of either the set 'GPSSENTITY' or 'ENTITY'. A noun phrase (NOUNPH) which is in the set (denoted by \$) 'GPSSENTITY', such as "the storage", results in a STATPH segment containing the appropriate code for that entity in attribute eight. Occurrence of the STATPH segment in the context "print STATPH statistics." results in the creation of a PRINTPH segment which

produces the appropriate calls to SETIND and SIMOUT to output the block or line of statistics. The calling parameters used for SETIND are the values of attributes eight and nine of the PRINTPH segment. These attributes and values are copied directly from the STATPH segment.

STATPH segments are also produced by noun phrases in the sets 'STATENTITY' ('stationary entity') and 'MOBENTITY' ('mobile entity'). These instances, such as "the pump" or "the car" in the proper context result in the production of the single lines of output corresponding to stationary entities or the table statistics for mobile entities. A series of two noun phrases, the first of which is in the set 'ENTITY' (either stationary or mobile) followed by a noun phrase in the set 'GPSSENTITY' (storage, queue, or table) also results in the creation of a STATPH. In this case, attribute eight is set to the code of the GPSS entity (accessed via the second noun phrase), and attribute nine is set to the identification number of the entity (via the first noun phrase). These parameters are then used in the SETIND routine to indicate the pertinent line of statistics to be produced by SIMOUT. A comparison of the rules and the general format described earlier provides an insight into the way in which these commands are presently handled.

4. Interrogating the Simulation Results

Additional rules were also incorporated to allow the user to ask questions about specific results of the simulation. With this added

capability, total, partial, or individual statistics are readily available to the user. Presently acceptable questions are of the form:

WHAT IS {THE} [SNA name] [OF] {THE} [Stationary Entity] ?
SNA [AT] Mobile Entity

The SNA's currently in use are those which correspond with the individual statistical elements produced in the GPSS-like printout. They are listed in the named record definitions in Appendix D (beginning with 'SC'). The character string attribute (CHARS) of each of these records serves as a natural language "SNA name" which can also be used in most instances. In utilizing the question form above, the user again must ensure compatibility between the SNA (or SNA name) and the type of entity statistic desired.

The two questions, "What is the SR of the pump?" and "What is the average utilization of the pump?", are equivalent and produce a response with the appropriate number. The same is true of the questions "What is the TB of the trucks?" and "What is the mean transit time of the trucks?". The only SNA's available presently for the mobile entities are TB, TC, and TD, which are "mean transit time", "number of entries", and "standard deviation". Storage and queue individual results may be obtained by using the SNA's SC, SM, SR, SA, S, R, ST, Q, QA, QM, QC, QZ, QT, QX, or QP with their associated stationary entity. The English name of each of these SNA's is contained in the CHARS attribute of the corresponding named record in Appendix D.

The rules for recognizing SNA names are also contained in the decoding lexology. Individual rules exist for each allowable SNA name. For example, in the question, "What is the current line at the pump?", "current line" results in a noun phrase segment in the set 'Q'. This noun phrase segment is later processed by an encoding rule (QUEST2) which tests for this set relation. Satisfaction of the rule conditions result in a sentence segment with attribute eight containing the SNACODE of the desired statistic and attribute nine containing the IDNO of the mobile or stationary entity. Using the values of attributes eight and nine as calling parameters, SPSTAT is called to return the value of the desired statistic. The value returned is then used in the response to the user.

V. CONCLUSIONS AND RECOMMENDATIONS

The thesis objective has been met. The simulation routine and associated rules have been integrated into the existing NLPQ system to produce an initial version of an interactive simulation system. With this feature, the NLPQ user can perform, and control the mode of, the simulation for his specific problem by using natural language commands. The results of the simulation may be requested in several ways. A complete GPSS-like printout is available or the user may select those portions of the printout which are of interest in his specific problem. Blocks of statistics (storage, queue, table, etc.) or individual lines of the statistical output are readily accessible by the user in the latter instance. Specific statistical results may also be requested in a question-answer fashion. Using these additional features, the user can solve queuing system problems in an interactive manner through natural language dialogue with the system.

Recommendations for further research include expansion of the present question handling abilities to allow further interrogation of the simulation results in a less stringent manner. The ability to detect incompatibilities in input questions and commands and produce meaningful error analyses would enhance the present interaction with the user. Extension of the present features of the simulation routine

to include a larger subset of those functions performed by GPSS, combined with the necessary rule language additions, would provide the user with greater power in solving more complex problems. Further enlargement of the present rule modules to permit interactive ability in specifying table limits, run times, and other GPSS attributes, would enable greater specification and control of the simulation by the user. A means of handling situations in which aggregate statistics are desired (for example, the station statistics should reflect the aggregate statistics for the pump or pumps in the station), would also be of value in improving the usefulness of the system.

APPENDIX A

*** NLP MODIFICATION TO ALLOW ***
X VECTOR READ OR WRITE

C			XRDWR
	ENTRY	XRDWR	
501	J=J+1		
	IF (COL(J).EQ.BLANK) GO TO 501		
	IF (COL(J).EQ.COLON) CALL ERRORA(J,9,&550)		
	CALL COLECT(NAME)		
511	J=J+1		
	IF (COL(J).EQ.BLANK) GO TO 511		
	IF (COL(J).EQ.COLON) CALL ERRORA(J,9,&550)		
	CALL CONVRT(NUM)		
	IF (NUM.LT.1.OR.NUM.GT.14) CALL ERRORA(J,10,&550)		
	NUM4=NUM		
	IF (NAME.EQ.XRD) GO TO 531		
	IF (NAME.EQ.XWR) GO TO 541		
	CALL ERRORA(J,9,&550)		
531	REWIND NUM4		
	READ (NUM4) (X(I),I=1,MAXX)		
	RETURN		
541	REWIND NUM4		
	WRITE (NUM4) (X(I),I=1,MAXX)		
550	RETURN		
	END		

*** ADDITICNAL ROUTINES ***

C	3210	CALL SIMULT(TR6,CUT6,RTERM,WTERM)	SIMULT
		GO TO 1200	
C	3220	CALL SIMOUT(OUT6)	SIMOUT
		GO TO 1200	
C	3230	ROW=HVAL(A8,SEGMNT)	SETIND
		BIT=HVAL(A9,SEGMNT)	
		CALL SETIND(ROW,BIT)	
		GO TO 1200	
C	3240	SNA=HVAL(A8,SEGMNT)	SPSTAT
		IDT=HVAL(A9,SEGMNT)	
		CALL SPSTAT(SNA,IDT,VAL,IORD)	
		LX=LOC(A9,SEGMNT)	
		CEL=CELL(LX)	
		TYPE=1	
		ADDR=NEWCEL(CONSTR(ZERO,IORD,PVAL(1),PVAL(2)))	
		CELL(LX)=CEL	
		GO TO 1200	

*** MODIFIED OUTPUT ROUTINE ***

C
C

PROCESS 'OUTPUT'

```

201 N = HVAL(A11,SEGMNT)
   IF (N.GT.0) CALL OUTCHR(DZERO)
   N = N - 1
   IF (N.GT.0) CALL SKIP(N)
   KOL = HVAL(A12,SEGMNT)
   IF (KOL.GT.0) CALL SETJJ(KOL)
   NXTWRD = 0
   FWRD = LOC(A13,SEGMNT)
   IF (FWRD.EQ.0) GO TO 231
221 WORD = DVALUE(FWRD,NXTWRD)
   DO 225 I=8,64,8
       CHR = DOR(DLS(DRS(WORD,64-I),56),DZB)
225   IF (CHR.NE.DBLANK) CALL OUTCHR(CHR)
   IF (NXTWRD.NE.0) GO TO 221
231 LOCC = LOC(A14,SEGMNT)
   IF (LOCC.EQ.0) GO TO 241
   DCM=0
   CEL=CELL(LOCC)
   IF (TYPE.EQ.1) GO TO 253
   RN=ADDR
   GO TO 257
241 LOCC = LOC(A15,SEGMNT)
   IF (LOCC.EQ.0) GO TO 251
   DCM=1
   CEL=CELL(LOCC)
   RN=ADDR/1000.
   GO TO 257
251 LOCC=LOC(A16,SEGMNT)
   IF (LOCC.EQ.0) GO TO 131
   CEL=CELL(LOCC)
253 CEL=CELL(ADDR)
   DCM=ATTR
   IF (DCM.EQ.1) GO TO 255
   RN=I4
   GO TO 257
255 RN=R4
257 IF (RN.GE.0.) GO TO 261
   RN=-RN
259 CALL OUTCHR(DMINUS)
261 IF (RN.GT.1.0E10) GO TO 298
   SW=0
   DO 265 I=1,13
       II=10-I
       IF (DCM.EQ.0.AND.II.EQ.-1) GO TO 131
       M=RN/10.**II+0.0005
       IF (SW.EQ.0.AND.M.EQ.0.AND.II.GT.0) GO TO 265
       SW=1
       IF (DCM.EQ.1.AND.II.EQ.-1) CALL OUTCHR(DECPNT)
       CALL OUTCHR(DIGIT(M+1))
       RN=RN-M*10.**II
265 CONTINUE
   GO TO 131
298 WRITE (OUT6,299) RN
299 FORMAT (' NUMBER EXCEEDS 'OUTPUT' LIMIT. VALUE IS '
1,E13.7)
   CALL OUTCHR(DSTAR)
   CALL OUTCHR(DSTAR)
   CALL OUTCHR(DSTAR)
   GO TO 131
END

```


APPENDIX B

DICTIONARY OF VARIABLES USED

IN THE SIMULATION ROUTINE

ALTBLO	Alternate block number for TEST or GATE routines.
AVAIL	Amount of storage available in a given storage entity.
BASE	Value to which spread will be added in GENERATE and ADVANCE blocks to determine departure time from the FEC.
BITTS	Value of requested bit pattern returned from call to GBITS.
BLOCK	Pointer to word preceding block directory in X-vector (BLOCK=X(19)).
BYTE (*)	Logical*1 temporary variable (BYTE(1)=DWORD(1)).
CBLO	Pointer to current block being processed.
CKPNT	Check point used for following traces when debugging.
CLOCK	Absolute clock time (CLOCK=X(11)).
CLOCKB	Base clock time needed to compute relative clock time after RESET.
CLOCKR	Relative clock time.
COMVAL	Comparison value used in SELECT blocks.
CQUE	Pointer to current queue being processed.
CREATE	Time differential between current clock and time transaction is due to leave the FEC.

CSTO	Pointer to current storage being processed.
CTRA	Pointer to current transaction being processed.
DELAY	Time delay caused by ADVANCE block.
DONES	Real*8 mask consisting of all ones.
DTIME	Real*8 time differential between clock and time queue or storage last changed status.
DW*	See DWORD(*).
DWORD(*)	Real*8 temporary variable (DW*=DWORD(*)).
EPT	Variable which flags entry point at which SIMULT was entered.
ERR	Flag set when an error is encountered (ERR=X(30)).
ERRORZ	Subroutine called to output error message and set ERR flag.
FB	Variable which specifies which bits to set when calling SETIND or the first bit to set when calling PBITS.
FFW*	See FFWORD(*).
FFWORD(*)	Real*4 temporary variable (FFW*=FFWORD(*); FFWORD(1)=DWORD(1)).
FLAG	Temporary variable used as a logical flag when needed.
FOS(*)	Stack for evaluating floating point arguments (FOS(*)=OS(*)).
FREWDT	Width of frequency interval in a table entity.
FRNG	Number of frequency intervals in a table entity.
FTCEC	Pointer to the first transaction on the Current Events Chain (FTCEC=X(26)).

FTFEC	Pointer to the first transaction on the Future Events Chain (FTFEC=X(28)).
FTRA	Pointer to the first transaction on the list of unused transactions.
FUNCT	Pointer to the word preceding the function directory in the X-vector (FUNCT=X(17)).
FW*	See FWORD(*).
FWORD(*)	Integer*4 temporary variable (FW*=FWORD(*); FWORD(1)=DWORD(1)).
GBITS	Function which returns value of bits specified.
GIVEUP	Number of storage units being made available.
HIGH..	Ending entity number to be used in SELECT block search.
HW*	See HWORD(*).
HWORD(*)	Integer*2 temporary variable (HW*=HWORD(*); HWORD(1)=DWORD(1)).
IDT	Identification number of entity to be used when calling SPSTAT.
IMAX	Looping limit for outputting the OS stack when tracing argument evaluations.
INST	Pointer to current block element being processed.
INTDEC(*)	Indicates whether corresponding OS/FOS value is integer or floating point.
IORD	Indicates whether value returned by SPSTAT is integer or decimal.
JBL	Number of the block whose routine is being executed.
KDEX	Index used to keep count of number of arguments specified for a block.

KMODE	Special mode variable used to determine how a function is to be evaluated.
KOUNT	Counter used as index into function entity tables.
KXVAL	Integer*4 value of independent variable used in function computation.
KYVAL	Integer*4 value of dependent variable used in function computation.
LB	Specifies the last bit to be set in a call to PBITS.
LOW	Beginning entity number to be used in SELECT block search.
LTCEC	Pointer to the last transaction on the Current Events Chain (LTCEC=X(27)).
LTFEC	Pointer to the last transaction on the Future Events Chain (LTFEC=X(29)).
MAXCTS	Maximum contents of a storage or queue.
MAXX	Maximum number of X-vector elements.
MDFR	Modifier used in TEST, SELECT, GATE, and ASSIGN blocks.
MODE	Indicates whether simulation will begin in Simulate/Start, Reset, Clear, or Start mode (MODE=X(1)).
MRNG	Number of points in a function.
NBLO	Number of blocks (NBLO=X(18)).
NEWBDT	Block departure time of transaction being merged into the FEC.
NEWPRI	Priority of transaction being merged into the CEC.

NEXBDT	Block departure time of first transaction on the FEC.
NEXBLO	Next block number transaction is to enter.
NFUNCT	Number of functions (NFUNCT=X(16)).
NOUPD	Indicator used to merge a transaction into the CEC without updating the clock.
NPAR	Number of transaction parameters (NPAR=X(24)).
NQUE	Number of queues (NQUE=X(14)).
NSAV	Number of savevalues (NSAV=X(22)).
NSTO	Number of storages (NSTO=X(12)).
NTAB	Number of tables (NTAB=X(20)).
NVAR	Number of GPSS-type variables (NVAR=X(31)).
OLDBDT	Block departure time of transaction being checked in the FEC.
OLDPRI	Priority of transaction being checked in the CEC.
OPCODE	Operation code indicating type of block.
OS(*)	Stack for evaluating integer arguments (OS(*)=FOS(*)).
OUTX	Specifies optional output file for X-vector (defaults to 0 (no output)).
OUT6	Specifies optional output file for traces and SIMOUT output (defaults to 6 (terminal)).
PBITS	Subroutine called to turn on specified bits in an indicator.
PCONTS	Present contents of a queue.

PNTA	Pointer to last entry on return address stack.
PNTB	Pointer to last evaluated argument on OS/FOS stack.
PNTF	Pointer to the first element of the frequency intervals for a table entity.
PNTS	Pointer to the first element of the slope values of the current function entity.
PNTT	Pointer to the transaction being compared with the current transaction.
PNTX	Pointer to the first element of the independent variable values of the current function entity.
PNTY	Pointer to the first element of the function values of the current function entity.
PR(*)	Temporary vector to save integer values in the OS stack for output while tracing.
QUE	Pointer to the word preceding the queue directory in the X-vector (QUE=X(15)).
RAS	Return address stack used to temporarily hold the location of the next arguments to be evaluated.
RN	Random number.
ROW	Argument to SETIND which specifies which status switch row is being set.
RR*	Temporary real*4 variables used primarily to save values for immediate output.
RSLOPE	Slope used in function evaluation.
RTERM	Specifies optional input file for entering optional data (defaults to 5 (terminal)).
RVAL	Value to be returned by SPSTAT if real*4 result is required.

RX(*)	Variable used to manipulate real*4 values in the X-vector (RX(*)=X(*)).
RXVAL	Real*4 value of the independent variable used in function computation.
RYVAL	Real*4 value of the dependent variable used in function computation.
SAVE	Pointer to the word preceding savevalue locations in the X-vector (SAVE=X(23)).
SCFLAG	Status change flag. Restarts scan at the beginning of the CEC when on.
SE	Pointer to the top of the pushdown chain for storage empty condition.
SEED(*)	Seed used in random number generation (SEED(1-8)=X(3-10)).
SETIND	Entry point. Called to set status switch indicators.
SF	Pointer to top of pushdown chain for storage full condition.
SIMOUT	Entry point. Called to output GPSS-like statistics.
SNA	Specifies the Standard Numerical Attribute to be evaluated on call to SPSTAT.
SNE	Pointer to the top of the pushdown chain for storage not empty condition.
SNF	Pointer to the top of the pushdown chain for storage not full condition.
SPSTAT	Entry point. Called to output special statistics (individual SNA's).
SRED	Pointer to the top of the pushdown chain for storage reduction in contents condition.

SSIND	Scan status indicator. True if the transaction is in an active status on the CEC.
STATSW(*)	Vector containing status switches which indicate which statistics are to be output by the SIMOUT routine.
STO	Pointer to the word preceding the storage directory in the X-vector (STO=X(13)).
TAB	Pointer to the word preceding the table directory in the X-vector (TAB=X(21)).
TASGN	The value being assigned in an ASSIGN block.
TBLNO	Number of table being tabulated.
TBLO	Temporary variable used to save the pointer to the current block.
TCNT	Temporary location for saved termination count in TERMINATE block.
TCTRA	Temporary variable for saving the pointer to the current transaction.
TEMP*	Temporary integer*4 variable.
TEST	Value of pointers associated with the delay chains.
TPAR	Parameter being assigned a value in an ASSIGN block.
TRACE	Logical switch for tracing simulation mode.
TRAN	Pointer to the word preceding the transaction entities (TRAN=X(25)).
TRARG	Logical switch for tracing argument evaluation.
TRBLO	Logical switch for tracing block routines.
TRCNT	Indicates the transaction number being assigned a new transaction.

TRMCNT	Termination count (TRMCNT=X(2)).
TRSCN	Logical switch for tracing scan procedure.
TRSIZE	Indicates the size of the X-vector printout.
TRS1	Logical switch for tracing chain manipulations.
TRTN	Temporary variable used to save the ZRTN value.
TRUPD	Logical switch for tracing update clock procedure.
TR6	Logical trace enable switch.
TTRA	Temporary variable used to save the pointer to the current transaction.
TYPARG	Indicates whether the search argument of a function entity is integer or floating point.
TYPX	Indicates whether the independent variable of a function entity is integer or floating point.
TYPY	Indicates whether the function values of a function entity are integer or floating point.
ULLI	Indicates the upper limit of the lowest frequency interval for a table entity.
UNITS	Number of units entering or leaving a queue.
USED	Current contents of a storage entity.
VAL	Value to be returned by SPSTAT as an integer*4.
VAR	Pointer to the word preceding the variable directory in the X-vector (VAR=X(32)).
WANT	Number of storage units required.
WRTN	Temporary variable used to save the ZRTN value.

WTERM	Specifies optional output file (defaults to 6 (terminal)).
WTFAC	Weighting factor utilized in TABULATE block. Defaults to one if not specified.
X(*)	Vector used for holding all storages, queues, tables, directories, etc.
ZRTN	Holds statement number for use in returning from various routines.

APPENDIX C

```

C
***** SIMULT *** SPSTAT *** SIMOUT *** SETIND *****
SUBROUTINE SIMULT(TR6,OUT6,RTERM,WTERM)
IMPLICIT INTEGER(A-Q,S-Z),REAL(R)
INTEGER*4 X(200),FWORD(4),RAS(10),OS(20),SEED(8),RTERM
INTEGER*2 HWORD(8),CTRA,FIRA,ZRTN,PNTA,PNTB,PNTT,INTDEC(20),PNTF,
-INST,PNTX,PNTY,PNTS,WRTN,TYPX,TYPY,CBLO,NEXBLO,MDFR,
- HW1,HW2,HW3,HW4,HW5,HW6,HW7,HW8,EPT
REAL*8 DWORD(2),DW1,DW2
REAL*4 FWORD(4),FOS(20),RX(1000),FFW1,FFW2,FFW3,FFW4
LOGICAL*1 BYTE(16),SSIND
LOGICAL TR6,TRACE,TRARG,TRBLO,TRUPD,TRSCN,TRULK,TRS1
COMMON /XVECTOR/ MAXX,X
EQUIVALENCE(DWORD,FWORD,FFWORD,HWORD,BYTE),(DWORD(1),DW1),
-(DWORD(2),DW2),(FWORD(2),FW2),(FWORD(3),FW3),(FWORD(4),FW4),
-(FFWORD(1),FFW1),(FFWORD(2),FFW2),(FFWORD(3),FFW3),
-(FFWORD(4),FFW4),
-(HWORD(1),HW1),(HWORD(2),HW2),(HWORD(3),HW3),(HWORD(4),HW4),
-(HWORD(5),HW5),(HWORD(6),HW6),(HWORD(7),HW7),(HWORD(8),HW8),
-(X(1),MODE),(X(2),TRMCNT),(X(3),SEED),(X(11),CLOCK),(X(12),NSTO),
-(X(13),STO),(X(14),NOUE),(X(15),QUE),(X(16),NFUNCTION),(X(17),NSAV),
-(X(18),NBLO),(X(19),BLOCK),(X(20),NTAB),(X(21),TAB),(X(22),NSAV),
-(X(23),SAVE),(X(24),NPAR),(X(25),TRAN),(X(26),FTCEC),
-(X(27),LTCEC),(X(28),FTFEC),(X(29),LTFEC),(X(30),ERR),
-(X(31),NVAR),(X(32),VAR),
-(X,RX),(OS,FOS)
NAMELIST/P/ TRACE,TRARG,TRBLO,TRUPD,TRSCN,TRULK,OUTX,TRSIZE,TRS1

```

```

C
C
***** SIMULT *****
INITIALIZE OPTIONAL TRACES

TRACE=.FALSE.
TRARG=.FALSE.
TRBLO=.FALSE.
TRUPD=.FALSE.
TRSCN=.FALSE.
TRULK=.FALSE.
TRS1=.FALSE.
OUTX=0
TRSIZE=MAXX
IF(TR6) WRITE(WTERM,9025)
IF(TR6) READ(RTERM,P)

SIM00010
SIM00020
SIM00030
SIM00040
SIM00050
SIM00060
SIM00070
SIM00080
SIM00090
SIM00100
SIM00110
SIM00120
SIM00130
SIM00140
SIM00150
SIM00160
SIM00170
SIM00180
SIM00190
SIM00200
SIM00210
SIM00220
SIM00230
SIM00240
SIM00250
SIM00260
SIM00270

SIM00340
SIM00350
SIM00360
SIM00370
SIM00380
SIM00390
SIM00400
SIM00420
SIM00430
SIM00440
SIM00450

```


SET ENTRY POINT AND BRANCH
TO APPROPRIATE MODE

SIM00460
SIM00470
SIM00480
SIM00490
SIM00510
SIM00530
SIM00540

*** BEGIN NEW SIMULATION ***
SLOPE COMPUTATION

SIM00550
SIM00570
SIM00580
SIM00590
SIM00600
SIM00610
SIM00620
SIM00630
SIM00640
SIM00650
SIM00660
SIM00670
SIM00680
SIM00690
SIM00700
SIM00710
SIM00720
SIM00730
SIM00735
SIM00740
SIM00750
SIM00800
SIM00810
SIM00820

CREATE CHAIN OF UNUSED TRANS

SIM00840
SIM00880
SIM00890
SIM00900
SIM00910
SIM00920
SIM00930
SIM00940
SIM00950
SIM00960
SIM00965

C

EPT=1
PNTT=0
IF (MODE.EQ.4) GO TO 400
IF (MODE.EQ.2) GO TO 60
CLOCK=0
CLOCKB=0
IF (MODE.EQ.3) GO TO 40

C

IF (TRACE) WRITE (OUT6,9003)

C

IF (TRACE) WRITE (OUT6,9002)

20

IF (K.GE.NFUNCT) GO TO 25

K=K+1

PNTF=X (FUNCT+K)

FW1=X (PNTF+5)

IF (HW1.GE.2) GO TO 25

MRNG=HW2

PNTX=PNTF+6

PNTY=PNTX+MRNG

PNTS=PNTY+MRNG

FW1=X (PNTX)

TYPX=HW1

TYPY=HW2

RX (PNTS+1)=0.0

DO 22 I=2,MRNG

RXVAL=X (PNTX+I)-X (PNTX+I-1)

IF (TYPX.EQ.1) RXVAL=RX (PNTX+I)-RX (PNTX+I-1)

RYVAL=X (PNTY+I)-X (PNTY+I-1)

IF (TYPY.EQ.1) RYVAL=RX (PNTY+I)-RX (PNTY+I-1)

RX (PNTS+I)=RYVAL/RXVAL

CONTINUE

GO TO 20

22

C

FTRA=TRAN
KK=MAXX-TRAN
KK=KK/(8+NPARG)-1
LL=TRAN

DO 27 I=1,KK

HW1=0

HW2=LL+8+NPARG

X (LL+1)=FW1

LL=LL+8+NPARG

CONTINUE

X (LL+1)=0

27

PLACE ALL GENERATE BLOCKS ON
FEC THEN JUMP TO SCAN

SIM01000
SIM01010
SIM01020
SIM01030
SIM01040
SIM01050
SIM01060
SIM01070
SIM01080
SIM01090
SIM01100
SIM01110
SIM01120
SIM01130

*** CLEAR ***

SIM01150
SIM01160
SIM01170
SIM01180
SIM01190

CLEAR STORAGES

SIM01300
SIM01310
SIM01320
SIM01325
SIM01340
SIM01330
SIM01350
SIM01360
SIM01370
SIM01380
SIM01390
SIM01400
SIM01410

CLEAR QUEUES

SIM01420
SIM01430
SIM01440
SIM01445
SIM01450
SIM01460
SIM01470
SIM01480
SIM01490
SIM01500

C
C
TRCNT=0
JBL=0
IF(JBL.EQ.NBLO) GOTO 400
JBL=JBL+1
CBLO=X(BLOCK+JBL)
FW1=X(CBLO+1)
OPCODE=HW1
IF(TRACE) WRITE(OUT6,9038) OPCODE
IF(OPCODE.NE.1) GOTO 30
INST=CBLO+2
ZRTN=32
GO TO 100
ZRTN=30
GOTO 200

C 40
IF(TRACE) WRITE(OUT6,9004)
FTCEC=0
LTCEC=0
FTFEC=0
LTFEC=0

C
IF(NSTO.EQ.0) GOTO 44
DO 43 I=1,NSTO
N=X(STO+I)
IF(N.EQ.0) GO TO 43
X(N+2)=X(N+2)+X(N+1)
X(N+1)=0
DW1=0.
X(N+3)=FW1
X(N+4)=FW2
X(N+5)=0
X(N+6)=0
X(N+7)=0
CONTINUE

43
C 44
IF(NQUE.EQ.0) GOTO 48
DO 47 I=1,NQUE
N=X(QUE+I)
IF(N.EQ.0) GO TO 47
X(N+1)=0
X(N+2)=0
X(N+3)=0
DW1=0.
X(N+4)=FW1
X(N+5)=FW2

SIM01510
SIM01520
SIM01530

CLEAR SAVEVALUES

SIM01540
SIM01550
SIM01560
SIM01570

CLEAR/RESET TABLES

SIM01580
SIM01590
SIM01600
SIM01605
SIM01610
SIM01620
SIM01630
SIM01640
SIM01645
SIM01650
SIM01660
SIM01670
SIM01680
SIM01690
SIM01695

SIM01700
SIM01710

CLEAR/RESET COMPLETED

*** RESET ***

SIM01740
SIM01750

RESET STORAGES

SIM01760
SIM01770
SIM01780
SIM01785
SIM01790
SIM01800
SIM01810
SIM01820
SIM01830
SIM01840
SIM01850

RESET QUEUES

SIM01860
SIM01870
SIM01880
SIM01885

```

47 X(N+6)=0
C  X(N+7)=0
   CONTINUE

48 IF(NSAV.EQ.0) GOTO 50
   DO 49 I=1,NSAV
   X(SAVE+I)=0
C  CONTINUE

49 IF(NTAB.EQ.0) GOTO 58
   DO 56 I=1,NTAB
   N=X(TAB+I)
   IF (N.EQ.0) GO TO 56
   RX(N+1)=0.0
   RX(N+2)=0.0
   RX(N+3)=0.0
   RX(N+4)=0.0
   X(N+5)=0
   RX(N+7)=0.0
   K=X(N+8)
   DO 55 J=1,K
   X(N+ 9+J)=0
C  CONTINUE
   CONTINUE

55 GOTO(59,400,25),MODE
56 CALL ERRORZ(603,1,&9999)
C

60 CLOCKB=CLOCK
C  IF(TRACE) WRITE(OUT6,9005)

   IF(NSTO.EQ.0) GOTO 63
   DO 62 I=1,NSTO
   N=X(STO+I)
   IF (N.EQ.0) GO TO 62
   DW1=0.
   X(N+3)=FW1
   X(N+4)=FW2
   X(N+5)=CLOCK
   X(N+6)=X(N+1)
   X(N+7)=X(N+1)
C  CONTINUE

62 IF(NQUE.EQ.0) GOTO 50
C  DO 67 I=1,NQUE
   N=X(QUE+I)
   IF (N.EQ.0) GO TO 67

```



```

X(N+1)=CLOCK
DW1=0.
X(N+4)=FW1
X(N+5)=FW2
X(N+2)=X(N+6)
X(N+3)=0
X(N+7)=X(N+6)
CONTINUE
GOTO 50

```

67

*** ARGUMENT EVALUATION ***

```

C 100      PNTA=0
          PNTB=0
          IF( TRARG ) WRITE(OUT6,9006)
1001      FW1=X(INST)
          IF(HW1.GE.0) GOTO 196
          IF(HW2) 1002,1004,1003
1002      HW2=-HW2
          IF(HW2.GT.NPAR) CALL ERRORZ(492,1,&9999)
          IF(HW2.LE.0) CALL ERRORZ(492,2,&9999)
          HW2=X(CTRA+8+HW2)
1003      HW1=-HW1
1006      GOTO (101,102,103,104,105,106,107,108,109,
-110,111,112,113,114,115,116,117,118,119,
-120,121,122,123,124,125,126,127,128,129,
-130,199,199,133),HW1
1004      IF(HW1.EQ.-33) GOTO 133
          IF(HW1.EQ.-23) GOTO 123
          IF(HW1.EQ.-25) GOTO 125
          FW2=X(CBLO+1)
          IF(HW3.NE.21.OR.PNTB.NE.4) GOTO 1005
          HW2=1
          GOTO 1003
          CALL ERRORZ(45,0,&9999)
C 1005
101      FW1=X(CTRA+8+HW2)
          GO TO 190
C 102      FW1=X(CTRA+5)
          HW1=0
          GOTO 190
C 103      FW1=CLOCK-X(CTRA+4)
          GOTO 190
C 104      FW1=CLOCK-X(CTRA+8+HW2)
          GOTO 190
C

```

SIM01890
SIM01900
SIM01910
SIM01920
SIM01930
SIM01940
SIM01950
SIM01960
SIM01970

SIM01990
SIM02000
SIM02010
SIM02020
SIM02030
SIM02040
SIM02050
SIM02060
SIM02070
SIM02080
SIM02090
SIM02100
SIM02110
SIM02120
SIM02130
SIM02140
SIM02150
SIM02160
SIM02170
SIM02180
SIM02190
SIM02200
SIM02210

SIM02230
SIM02240

SIM02260
SIM02270
SIM02280

SIM02300
SIM02310

SIM02330
SIM02340

P

PR

M1

MP

SC

105	IF(HW2.GT.NSTO) CALL ERRORZ(499,0,&9999)	SIM02360
	N=X(STO+HW2)	SIM02370
	FW1=X(N+6)	SIM02380
	GOTO 190	SIM02390
C 106		
	IF(HW2.GT.NSTO) CALL ERRORZ(499,0,&9999) SM	SIM02410
	N=X(STO+HW2)	SIM02420
	FW1=X(N+7)	SIM02430
	GOTO 190	SIM02440
C 107		
	IF(HW2.GT.NSTO) CALL ERRORZ(499,0,&9999) ^{SR}	SIM02460
	N=X(STO+HW2)	SIM02470
	FW3=X(N+3)	SIM02480
	FW4=X(N+4)	SIM02490
	FW1=X(N+1)+X(N+2)	SIM02500
	FW1=FW1÷CLOCK	SIM02510
	RVAL=DW2/FW1	SIM02520
	FW1=(DW2/FW1)*1000	SIM02530
	GOTO 190	SIM02540
C 108		
	IF(HW2.GT.NSTO) CALL ERRORZ(499,0,&9999) ^{SA}	SIM02560
	N=X(STO+HW2)	SIM02570
	FW3=X(N+3)	SIM02580
	FW4=X(N+4)	SIM02590
	FW1=DW2/CLOCK	SIM02600
	RVAL=DW2/CLOCK	SIM02610
	GOTO 190	SIM02620
C 109		
	IF(HW2.GT.NSTO) CALL ERRORZ(499,0,&9999) ^S	SIM02640
	N=X(STO+HW2)	SIM02650
	FW1=X(N+1)	SIM02660
	GOTO 190	SIM02670
C 110		
	IF(HW2.GT.NSTO) CALL ERRORZ(499,0,&9999) ^R	SIM02690
	N=X(STO+HW2)	SIM02700
	FW1=X(N+2)	SIM02710
	GOTO 190	SIM02720
C 111		
	IF(HW2.GT.NSTO) CALL ERRORZ(499,0,&9999) ST	SIM02740
	N=X(STO+HW2)	SIM02750
	FW3=X(N+3)	SIM02760
	FW4=X(N+4)	SIM02770
	FW1=0	SIM02780
	IF (X(N+6).EQ.0) GO TO 190	SIM02790
	FW1=DW2/X(N+6)	SIM02800
	RVAL=DW2/X(N+6)	SIM02810
	GOTO 190	SIM02820
C		

Q


```

112 IF(HW2.GT.NQUE) CALL ERRORZ(500,0,&9999)
   N=X(QUE+HW2)
   FW1=X(N+6)
   GOTO 190
C 113 IF(HW2.GT.NQUE) CALL ERRORZ(500,0,&9999)
   N=X(QUE+HW2)
   FW3=X(N+4)
   FW4=X(N+5)
   FW1=DW2/CLOCK
   RVAL=DW2/CLOCK
   GOTO 190
C 114 IF(HW2.GT.NQUE) CALL ERRORZ(500,0,&9999)
   N=X(QUE+HW2)
   FW1=X(N+7)
   GOTO 190
C 115 IF(HW2.GT.NQUE) CALL ERRORZ(500,0,&9999)
   N=X(QUE+HW2)
   FW1=X(N+2)
   GOTO 190
C 116 IF(HW2.GT.NQUE) CALL ERRORZ(500,0,&9999)
   N=X(QUE+HW2)
   FW1=X(N+3)
   GOTO 190
C 117 IF(HW2.GT.NQUE) CALL ERRORZ(500,0,&9999)
   N=X(QUE+HW2)
   FW3=X(N+4)
   FW4=X(N+5)
   FW1=0
   IF (X(N+2).EQ.0) GO TO 190
   FW1=DW2/X(N+2)
   RVAL=DW2/X(N+2)
   GOTO 190
C 118 IF(HW2.GT.NQUE) CALL ERRORZ(500,0,&9999)
   N=X(QUE+HW2)
   FW3=X(N+4)
   FW4=X(N+5)
   FW1=X(N+2)-X(N+3)
   IF (FW1.EQ.0) GO TO 190
   RVAL=DW2/FW1
   FW1=DW2/FW1
   GOTO 190
C

```

TB

```

SIM02840
SIM02850
SIM02860
SIM02870
SIM02890
SIM02900
SIM02910
SIM02920
SIM02930
SIM02940
SIM02950
SIM02970
SIM02980
SIM02990
SIM03000
SIM03020
SIM03030
SIM03040
SIM03050
SIM03070
SIM03080
SIM03090
SIM03100
SIM03120
SIM03130
SIM03140
SIM03150
SIM03160
SIM03170
SIM03180
SIM03190
SIM03200
SIM03220
SIM03230
SIM03240
SIM03250
SIM03260
SIM03270
SIM03280
SIM03290
SIM03300

```


119	IF(HW2.GT.NTAB) CALL ERRORZ(435,0,&9999)	SIM03320
	N=X(TAB+HW2)	SIM03330
	FFW1=0.	SIM03340
	IF (X(N+5).EQ.0) GO TO 190	SIM03350
	FFW1=RX(N+1)/X(N+5)	SIM03360
	RVAL=FFW1	SIM03370
	GO TO 190	SIM03380
C 120	IF(HW2.GT.NTAB) CALL ERRORZ(435,0,&9999)	SIM03400
	N=X(TAB+HW2)	SIM03410
	FW1=X(N+5)	SIM03420
	GO TO 190	SIM03430
C 121	IF(HW2.GT.NTAB) CALL ERRORZ(435,0,&9999)	SIM03450
	N=X(TAB+HW2)	SIM03460
	FFW1=0.	SIM03470
	IF (X(N+5).LE.1) GO TO 190	SIM03480
	FFW1=RX(N+2)-((RX(N+1))*2)/X(N+5)	SIM03490
	FFW1=SQRT((FFW1/(X(N+5)-1))	SIM03500
	RVAL=FFW1	SIM03510
	GO TO 190	SIM03520
C 122	IF(HW2.GT.NFUNCT) CALL ERRORZ(507,0,&9999)	SIM03540
	PNTA=PNTA+1	SIM03550
	RAS(PNTA)=INST+1	SIM03560
	INST=X(FUNCT+HW2)+1	SIM03570
	GO TO 1001	SIM03580
123	FW1=X(INST+3)	SIM03590
	MRNG=HW2	SIM03600
	PNTX=INST+4	SIM03610
	PNTY=PNTX	SIM03620
	IF(HW1.LE.3) PNTY=PNTY+MRNG	SIM03630
	PNTS=PNTY+MRNG	SIM03640
	KARG=OS(PNTB)	SIM03650
	RKARG=FOS(PNTB)	SIM03660
	TPARG=INTDEC(PNTB)	SIM03670
	KMCDE=2*(HW1-1)+TPARG+1	SIM03680
	FW2=X(INST+4)	SIM03690
	TPX=HW3	SIM03700
	TPY=HW4	SIM03710
	IF(KMODE.GT.7) GO TO 1235	SIM03720
	KCUNT=0	SIM03730
	IF(TYPX.EQ.0) GO TO 1232	SIM03740
	IF(TYPARG.EQ.0) RKARG=KARG	SIM03750
	DO 1231 I=1,MRNG	SIM03760
	KOUNT=KOUNT+1	SIM03770
1231	IF(RKARG.LE.RX(PNTX+KOUNT)) GO TO 1234	SIM03780
	CONTINUE	SIM03790


```

1232 GOTO 1234
      IF(TYPARG.EQ.1) KARG=RKARG
DO 1233 I=1,MRNG
      KOUNT=KOUNT+1
      IF(KARG.LE.X(PNTX+KOUNT)) GOTO 1234
CONTINUE
1233 IF(KMODE.LE.2) GOTO 1237
1234 KARG=KOUNT
      TYPARG=0
      KMCDE=KMCDE+4
1235 IF(TYPARG.EQ.1) KARG=RKARG
      IF(KARG.GT.MRNG) CALL ERRORZ(509,0,&9999)
      IF(KARG.LT.1) CALL ERRORZ(509,0,&9999)
      IF(KMODE.GE.9) GOTO 1236
      OS(PNTB)=X(PNTY+KARG)
      INTDEC(PNTB)=0
      GOTO 1239
1236 X(INST+1)=X(PNTY+KARG)
      PNTB=PNTB-1
      INST=INST+1
      GOTO 1001
1237 KXVAL=X(PNTX+KOUNT-1)
      RXVAL=RX(PNTX+KOUNT-1)
      KYVAL=X(PNTY+KOUNT-1)
      RYVAL=RX(PNTY+KOUNT-1)
      RSLOPE=RX(PNTS+KOUNT)
      INTDEC(PNTB)=1
      KOUNT=2
      IF(KOUNT.EQ.1) GOTO 1238
      IF(KMODE.EQ.1)
      FFW1=RX(PNTX+MRNG)
      IF(RKARG.GT.FFW1) RKARG=FFW1
      RXVAL=RKARG-RXVAL
      RXVAL=RSLOPE-RXVAL
      FOS(PNTB)=RYVAL+RXVAL
      IF(TYPY.EQ.0) FOS(PNTB)=KYVAL+RXVAL
      GOTO 1239
1238 FFW1=X(PNTX+MRNG)
      IF(KARG.GT.FFW1) KARG=FWW1
      KXVAL=KARG-KXVAL
      RXVAL=RSLOPE-KXVAL
      FOS(PNTB)=RYVAL+RXVAL
      IF(TYPY.EQ.0) FOS(PNTB)=KYVAL+RXVAL
1239 INST=PNTA-1
      PNTA=PNTA-1
      GOTO 1001
C
124 IF(HW2.GT.NVAR) CALL ERRORZ(514,0,&9999)
      PNTA=PNTA+1

```

```

SIM03800
SIM03810
SIM03820
SIM03830
SIM03840
SIM03850
SIM03860
SIM03870
SIM03880
SIM03890
SIM03900
SIM03910
SIM03920
SIM03930
SIM03940
SIM03950
SIM03960
SIM03970
SIM03980
SIM03990
SIM04000
SIM04010
SIM04020
SIM04030
SIM04040
SIM04050
SIM04060
SIM04070
SIM04080
SIM04090
SIM04100
SIM04110
SIM04120
SIM04130
SIM04140
SIM04150
SIM04160
SIM04170
SIM04180
SIM04190
SIM04200
SIM04210
SIM04220
SIM04230
SIM04240
SIM04260
SIM04270

```



```

125  RAS(PNTA)=INST+1
      INST=X(VAR+HW2)+1
      GOTO 1001
      INST=RAS(PNTA)
      PNTA=PNTA-1
      IF(INTDEC(PNTB).EQ.0) GOTO 1001
      OS(PNTB)=FOS(PNTB)
      INTDEC(PNTB)=0
      GOTO 1001

C 126  IF(HW2.GT.8) CALL ERRORZ(45,0,&9999)
      SEED(HW2)=SEED(HW2)*65539
      RN=0.5+SEED(HW2)*0.2328306E-9
      IF(PNTA.EQ.0) GOTO 1261
      FW1=X(RAS(PNTA)-1)
      IF(HW1.NE.-22) GOTO 1261
      PNTB=PNTB+1
      FOS(PNTB)=RN
      INTDEC(PNTB)=1
      GOTO 191
      FW1=100J.*RN
      GOTO 190

C 127  FW1=CLOCK-CLOCKB
      GOTO 190

C 128  FW1=X(SAVE+HW2)
      GOTO 190

C 129  IF (HW2.GT.NQUE) CALL ERRORZ(500,0,&9999)
      N=X(QUE+HW2)
      RVAL=0.
      IF (X(N+2).NE.0) RVAL=(100.*X(N+3))/X(N+2)
      GOTO 190

C 130  PNTB=PNTB-1
      K=4*(HW2-1)+2*INTDEC(PNTB)+INTDEC(PNTB+1)+1
      GOTO (1301,1302,1303,1304,1305,1306,1307,1308,1309,1310,1311,
-1312,1313,1314,1315,1316,1317,1318,1319,1320),K
      OS(PNTB)=OS(PNTB)+OS(PNTB+1)
      GOTO 1321
      FOS(PNTB)=OS(PNTB)+FOS(PNTB+1)
      GOTO 1322
      FOS(PNTB)=FOS(PNTB)+OS(PNTB+1)
      GOTO 1322
      FOS(PNTB)=FOS(PNTB)+FOS(PNTB+1)
      GOTO 1322
      OS(PNTB)=OS(PNTB)--OS(PNTB+1)

```

```

SIM04280
SIM04290
SIM04300
SIM04310
SIM04320
SIM04330
SIM04340
SIM04350
SIM04360
SIM04380
SIM04390
SIM04400
SIM04410
SIM04420
SIM04430
SIM04440
SIM04450
SIM04460
SIM04470
SIM04480
SIM04490
SIM04510
SIM04520
SIM04540
SIM04550
SIM04570
SIM04580
SIM04590
SIM04600
SIM04610
SIM04630
SIM04640
SIM04650
SIM04660
SIM04670
SIM04680
SIM04690
SIM04700
SIM04710
SIM04720
SIM04730
SIM04740
SIM04750

```


1306	GOTO 1321	FOS(PNTB)=OS(PNTB)-FOS(PNTB+1)	SIM04760
1307	GOTO 1322	FOS(PNTB)=FOS(PNTB)-OS(PNTB+1)	SIM04770
1308	GOTO 1322	FOS(PNTB)=FOS(PNTB)-FOS(PNTB+1)	SIM04780
1309	GOTO 1322	OS(PNTB)=OS(PNTB)*OS(PNTB+1)	SIM04790
1310	GOTO 1321	FOS(PNTB)=OS(PNTB)*FOS(PNTB+1)	SIM04800
1311	GOTO 1322	FOS(PNTB)=FOS(PNTB)*OS(PNTB+1)	SIM04810
1312	GOTO 1322	FOS(PNTB)=FOS(PNTB)*FOS(PNTB+1)	SIM04820
1313	GOTO 1322	OS(PNTB)=OS(PNTB)/OS(PNTB+1)	SIM04830
1314	GOTO 1321	FOS(PNTB)=OS(PNTB)/FOS(PNTB+1)	SIM04840
1315	GOTO 1322	FOS(PNTB)=FOS(PNTB)/OS(PNTB+1)	SIM04850
1316	GOTO 1322	FOS(PNTB)=FOS(PNTB)/FOS(PNTB+1)	SIM04860
1317	GOTO 1322	OS(PNTB)=MOD(OS(PNTB),OS(PNTB+1))	SIM04870
1318	GOTO 1321	FOS(PNTB)=AMOD(FLOAT(OS(PNTB)),FOS(PNTB+1))	SIM04880
1319	GOTO 1322	FOS(PNTB)=AMOD(FOS(PNTB),FLOAT(OS(PNTB+1)))	SIM04890
1320	GOTO 1322	FOS(PNTB)=AMOD(FOS(PNTB),FOS(PNTB+1))	SIM04900
1321	INTDEC(PNTB)=0		SIM04910
1322	GOTO 191		SIM04920
		RETURN FROM ARG. EVALUATION	SIM04930
C 133	MDFR=HW2		SIM04940
	I=2		SIM04950
1331	KDEX=PNTB		SIM04960
	IF(I.GT.PNTB) GOTO 1333		SIM04970
	IF(INTDEC(PNTB).EQ.0) GOTO 1332		SIM04980
	IF(I.EQ.2.AND.(OPCODE.EQ.1.OR.OPCODE.EQ.2)) GOTO 1332		SIM04990
	OS(I)=FOS(I)		SIM05000
	INTDEC(I)=0		SIM05010
	I=I+1		SIM05020
1332	GOTO 1331		SIM05030
1333	IF(PNTB.EQ.7) GOTO 1334		SIM05040
	PNTB=PNTB+1		SIM05050
			SIM05060
			SIM05070
			SIM05080
			SIM05090
			SIM05100
			SIM05120
			SIM05130
			SIM05140
			SIM05150
			SIM05160
			SIM05170
			SIM05180
			SIM05190
			SIM05200
			SIM05210
			SIM05220
			SIM05230


```

OS(PNTB)=0
INTDEC(PNTB)=0
GOTO 1333
IF(.NOT.TRACE) GOTO 1337
INTEGER*4 PR(7)
IMAX=1
DO 1335 I=1,7
PR(I)=0
IF(INTDEC(I).EQ.0) PR(I)=OS(I)
IF(PR(I).NE.0) IMAX=I
CONTINUE
1335 IF(TRACE) WRITE(OUT6,9015) CBLO,OPCODE,MDFR,(PR(I),I=1,IMAX)
DO 1336 I=1,7
IF(INTDEC(I).EQ.1) WRITE(OUT6,9016) I,FOS(I)
CONTINUE
1336 IF(ZRTN.EQ.32) GOTO 32
1337 IF(ZRTN.EQ.2316) GOTO 2316
IF(ZRTN.EQ.4004) GOTO 4004
CALL ERRORZ(603,6,&9999)

C 190 IF (EPT.EQ.2) GO TO 5950
C 196 PNTB=PNTB+1
OS(PNTB)=FW1
INTDEC(PNTB)=0
191 INST=INST+1
GOTO 1001
199 CALL ERRORZ(45,0,&9999)

*** BLOCK ROUTINES ***
C 200 IF(TRBLO) WRITE(OUT6,9007) OPCODE,CBLO
GOTO (201,202,203,204,205,206,207,208,209,210,211,212,213,299,
-299,216,299,299,299,299,221),OPCODE
GENERATE
C 201 IF(KDEX.GE.4) GOTO 2014
CTRA=CTRA+1
FW1=X(CTRA+1)
CTRA=FW2
K=8+NPAP
DO 2015 I=1,K
X(CTRA+I)=0
CONTINUE
2015 IF(HW2.EQ.0) CALL ERRORZ(468,0,&9999)
IF(CTRA=0) WRITE(OUT6,9036) FTRA
HW1=0
HW2=JBL
X(CTRA+1)=FW1
TRCNT=TRCNT+1

```

```

SIM05240
SIM05250
SIM05260
SIM05270
SIM05280
SIM05290
SIM05300
SIM05310
SIM05320
SIM05330
SIM05340
SIM05350
SIM05360
SIM05370
SIM05380
SIM05390
SIM05400
SIM05410
SIM05420
SIM05440
SIM05460
SIM05470
SIM05480
SIM05490
SIM05500
SIM05510
SIM05530
SIM05540
SIM05550
SIM05570
SIM05580
SIM05590
SIM05600
SIM05610
SIM05620
SIM05630
SIM05640
SIM05650
SIM05660
SIM05670
SIM05680
SIM05690
SIM05700

```



```

HW1=MDFR
HW2=TRCNT
X(CTRA+8)=FW1
HW1=JBL+1
HW2=OS(5)
X(CTRA+5)=FW1
BYTE(1)=.TRUE.
BYTE(2)=.TRUE.
HW2=0
X(CTRA+6)=FW1
FW1=X(CBLO+3)
IF(KDEX.GE.2.AND.HW1.EQ.-22) GOTO 2011
IF(OS(2).NE.0) GOTO 2013
CREATE=OS(1)
IF(INDEC(1).EQ.1) CREATE=FOS(1)
GOTO 2012
2013 SEED(1)=SEED(1)*65539
RN=0.5+SEED(1)*0.2328306E-9
BASE=OS(1)-OS(2)
2019 IF(BASE.LT.0) CALL ERRORZ(505,0,&9999)
CREATE=BASE+(2*OS(2)*RN+0.5)
GO TO 2012
2011 BASE=OS(1)
CREATE=BASE*OS(2)
IF(INTEC(2).EQ.1) CREATE=BASE*FOS(2)
2012 X(CTRA+3)=CLOCK+CREATE
X(CTRA+4)=X(CTRA+3)
IF(CBLO) WRITE(OUT6,9021) X(CTRA+3)
IF(CREATE.NE.0) GOTO 3104
NOUPD=1
X(CTRA+2)=X(CTRA+1)
GOTO 3001
2014 NOUPD=0
IF(ZRIN.EQ.2314) GOTO 2314
IF(ZRTN.EQ.30) GOTO 30
CALL ERRORZ(603,7,&9999)

C ADVANCE
202 FW1=X(CTRA+5)
CBLO=X(CBLO+HW1)
FW1=X(CBLO+3)
IF(HW1.EQ.-22) GOTO 2021
SEED(1)=SEED(1)*65539
RN=0.5+SEED(1)*0.2328306E-9
BASE=OS(1)-OS(2)
IF(BASE.LT.0) CALL ERRORZ(505,0,&9999)
DELAY=BASE+(2*OS(2)*RN+0.5)
GOTO 2022
2021 BASE=OS(1)

```

SIM05710
SIM05720
SIM05730
SIM05740
SIM05750
SIM05760
SIM05770
SIM05780
SIM05790
SIM05800
SIM05810
SIM05820
SIM05830
SIM05840
SIM05850
SIM05860
SIM05870
SIM05880
SIM05890
SIM05900
SIM05910
SIM05920
SIM05930
SIM05940
SIM05950
SIM05960
SIM05970
SIM05980
SIM05990
SIM06000
SIM06010
SIM06020
SIM06030
SIM06040
SIM06050
SIM06060

SIM06080
SIM06090
SIM06100
SIM06110
SIM06120
SIM06130
SIM06140
SIM06150
SIM06160
SIM06170
SIM06180

2022	IF(INTDEC(1).EQ.1) BASE=FOS(1)	SIM06190
	IF(INTDEC(2).EQ.1) DELAY=BASE#FOS(2)	SIM06200
	X(CTRA+3)=CLOCK+DELAY	SIM06210
	GOTO 230	SIM06220
C 203		
	NEXBLO=OS(2)	SIM06240
	IF(NEXBLO.EQ.0) NEXBLO=OS(1)	SIM06250
	IF(NEXBLO.LT.1.OR.NEXBLO.GT.NBLO) CALL ERRORZ(603,8,&9999)	SIM06260
	GOTO 231	SIM06270
C 204		
	TPAR=OS(1)	SIM06290
	TASGN=OS(2)	SIM06300
	IF(TPAR.GT.NPAR) CALL ERRORZ(492,0,&9999)	SIM06310
	IF(KDEX.LE.2) GOTO 2042	SIM06320
	TASGN=TASGN#OS(3)	SIM06330
	IF(INTDEC(3).EQ.1) TASGN=TASGN#FOS(3)	SIM06340
2042	IF(MDFR-2) 2043,2044,2045	SIM06350
2043	X(CTRA+8+TPAR)=X(CTRA+8+TPAR)-TASGN	SIM06360
	GOTO 2046	SIM06370
2044	X(CTRA+8+TPAR)=TASGN	SIM06380
	GOTO 2046	SIM06390
2045	X(CTRA+8+TPAR)=X(CTRA+8+TPAR)+TASGN	SIM06400
2046	GOTO 230	SIM06410
C 205		
	IF(OS(1).GT.NPAR) CALL ERRORZ(492,0,&9999)	SIM06430
	IF(OS(1).EQ.0) GOTO 2051	SIM06440
	X(CTRA+8+OS(1))=CLOCK	SIM06450
	GOTO 230	SIM06460
2051	X(CTRA+4)=CLOCK	SIM06470
	GOTO 230	SIM06480
C 206		
	IF(OS(1).GT.NSTO) CALL ERRORZ(499,0,&9999)	SIM06500
	CSTO=X(STO+OS(1))	SIM06510
	AVAIL=X(CSTO+2)	SIM06520
	WANT=OS(2)	SIM06530
	IF(WANT.EQ.0) WANT=1	SIM06540
	IF(WANT.GT.AVAIL) GOTO 2064	SIM06550
	USED=X(CSTO+1)	SIM06560
	PCONTS=USED	SIM06570
	USED=USED+WANT	SIM06580
	AVAIL=AVAIL-WANT	SIM06590
	MAXCTS=X(CSTO+7)	SIM06600
	IF(USED.GT.MAXCTS) X(CSTO+7)=USED	SIM06610
	X(CSTO+1)=USED	SIM06620
	X(CSTO+2)=AVAIL	SIM06630
	X(CSTO+6)=X(CSTO+6)+WANT	SIM06640
	DTIME=CLOCK-X(CSTO+5)	SIM06650
	X(CSTO+5)=CLOCK	SIM06660

FW1=X(CSTO+3)				SI M06670
FW2=X(CSTO+4)				SI M06680
DW1=DW1+DTIME*PCONTS				SI M06690
X(CSTO+3)=FW1				SI M06700
X(CSTO+4)=FW2				SI M06710
FW1=X(CSTO+8)				SI M06720
SF=HW1				SI M06730
SNE=HW2				SI M06740
IF(SF.EQ.0.OR.AVAIL.NE.0) GOTO 2063				SI M06750
TEST=SF				SI M06760
SF=0				SI M06770
IF(TEST.EQ.0) GOTO 2062				SI M06780
FW1=X(TEST+6)				SI M06790
BYTE(2)=.TRUE.				SI M06800
X(TEST+6)=FW1				SI M06810
FW1=X(TEST+1)				SI M06820
HW3=TEST				SI M06830
TEST=HW1				SI M06840
HW1=0				SI M06850
X(HW3+1)=FW1				SI M06860
GOTO 2061				SI M06870
SCFLAG=1				SI M06880
2062 IF(SNE.EQ.0.OR.USED.EQ.0) GOTO 230				SI M06890
2063 TEST=SNE				SI M06900
SNE=0				SI M06910
GOTO 2061				SI M06920
FW1=X(CTRA+1)				SI M06930
FW2=X(CSTO+10)				SI M06940
HW1=HW3				SI M06950
HW3=CTRA				SI M06960
X(CTRA+1)=FW1				SI M06970
X(CSTO+10)=FW2				SI M06980
FW1=X(CTRA+6)				SI M06990
BYTE(2)=.FALSE.				SI M07000
X(CTRA+6)=FW1				SI M07010
GOTO 4002				SI M07020
C 207				
IF(OS(1).GT.NSTO) CALL ERRORZ(499,0,&99999)				SI M07040
CSTO=X(STO+OS(1))				SI M07050
USED=X(CSTO+1)				SI M07060
PCONTS=USED				SI M07070
GIVEUP=OS(2)				SI M07080
IF(GIVEUP.EQ.0) GIVEUP=1				SI M07090
IF(GIVEUP.GT.USED) CALL ERRORZ(425,0,&99999)				SI M07100
AVAIL=X(CSTO+2)				SI M07110
USED=USED-GIVEUP				SI M07120
AVAIL=AVAIL+GIVEUP				SI M07130
X(CSTO+1)=USED				SI M07140

SIM07150
SIM07160
SIM07170
SIM07180
SIM07190
SIM07200
SIM07210
SIM07220
SIM07230
SIM07240
SIM07250
SIM07260
SIM07270
SIM07280
SIM07290
SIM07300
SIM07310
SIM07320
SIM07330
SIM07340
SIM07350
SIM07360
SIM07370
SIM07380
SIM07390
SIM07400
SIM07410
SIM07420
SIM07430
SIM07440
SIM07450
SIM07460
SIM07470
SIM07480
SIM07490

SIM07510
SIM07520
SIM07530
SIM07540
SIM07550
SIM07560
SIM07570
SIM07580
SIM07590
SIM07600
SIM07610
SIM07620

```

X(CSTO+2)=AVAIL
DTIME=CLOCK-X(CSTO+5)
X(CSTO+5)=CLOCK
FW1=X(CSTO+3)
FW2=X(CSTO+4)
DW1=DW1+DTIME*PCONTS
X(CSTO+3)=FW1
X(CSTO+4)=FW2
FW1=X(CSTO+9)
SE=HW1
SNF=HW2
FW1=X(CSTO+10)
SRED=HW1
IF(SE.EQ.0.OR.USED.GT.0) GOTO 2073
TEST=SE
SE=0
2071 IF(TEST.EQ.0) GOTO 2072
FW1=X(TEST+6)
BYTE(2)=.TRUE.
X(TEST+6)=FW1
FW1=X(TEST+1)
HW3=TEST
TEST=HW1
HW1=0
X(HW3+1)=FW1
GOTO 2071
SCFLAG=1
2072 IF(SNF.EQ.0.OR.AVAIL.EQ.0) GOTO 2074
2073 TEST=SNF
SNF=0
GOTO 2071
2074 IF(SRED.EQ.0.OR.GIVEUP.EQ.0) GOTO 230
TEST=SRED
SRED=0
GOTO 2071
C
208 IF(OS(1).GT.NQUE) CALL ERRORZ(500,0,&9999)
CQUE=X(CQUE+OS(1))
UNITS=OS(2)
IF(UNITS.EQ.0) UNITS=1
PCONTS=X(CQUE+6)
X(CQUE+2)=X(CQUE+2)+UNITS
X(CQUE+6)=PCONTS+UNITS
MAXCTS=X(CQUE+7)
IF(X(CQUE+6).GT.MAXCTS) X(CQUE+7)=X(CQUE+6)
DTIME=CLOCK-X(CQUE+1)
X(CQUE+1)=CLOCK
FW1=X(CQUE+4)

```


SIM07630
 SIM07640
 SIM07650
 SIM07660
 SIM07670
 SIM07680
 SIM07690
 SIM07700
 SIM07710

SIM07730
 SIM07740
 SIM07750
 SIM07760
 SIM07770
 SIM07780
 SIM07790
 SIM07800
 SIM07810
 SIM07820
 SIM07830
 SIM07840
 SIM07850
 SIM07860
 SIM07870
 SIM07880

SIM07900
 SIM07910
 SIM07920
 SIM07930
 SIM07940
 SIM07950
 SIM07960
 SIM07970
 SIM07980
 SIM07990
 SIM08000
 SIM08010
 SIM08020
 SIM08030
 SIM08040
 SIM08050
 SIM08060
 SIM08070
 SIM08080
 SIM08090
 SIM08092

```

FW2=X(CQUE+5)
DW1=DW1+DTIME#PCONTS
X(CQUE+4)=FW1
X(CQUE+5)=FW2
FW1=X(CTRA+6)
HW2=OS(1)
X(CTRA+6)=FW1
X(CTRA+7)=CLOCK
GOTO 230

C 209      IF(OS(1).GT.NQUE) CALL ERRORZ(500,0,&9999)
           CQUE=X(CQUE+OS(1))
           UNITS=OS(2)
           IF(UNITS.EQ.0) UNITS=1
           PCONTS=X(CQUE+6)
           IF(UNITS.GT.X(CQUE+6)) CALL ERRORZ(428,0,&9999)
           IF(X(CTRA+7).EQ.CLOCK) X(CQUE+3)=X(CQUE+3)+UNITS
           X(CQUE+6)=X(CQUE+6)-UNITS
           DTIME=CLOCK-X(CQUE+1)
           X(CQUE+1)=CLOCK
           FW1=X(CQUE+4)
           FW2=X(CQUE+5)
           DW1=DW1+DTIME#PCONTS
           X(CQUE+4)=FW1
           X(CQUE+5)=FW2
           GOTO 230
  
```

TERMINATE

```

C 210      IF(.NOT.TRBLO) GOTO 2105
           WRITE(OUT6,9041) TRMCNT
           KK=8+NPAR
           DO 2106 I=1, KK
           FW1=X(CTRA+1)
           WRITE(OUT6,9019) I,FW1,HW1,HW2
           CONTINUE
2106      FW1=X(CTRA+2)
2105      IF(HW1.EQ.0) GOTO 2102
           FW2=X(HW1+2)
           HW4=HW2
           X(HW1+2)=FW2
2101      IF(HW2.EQ.0) GOTO 2103
           FW2=X(HW2+2)
           HW3=HW1
           X(HW2+2)=FW2
           GOTO 2104
2102      FTCEC=HW2
2103      GOTO 2101
2104      LTCEC=HW1
           FW1=X(CTRA+1)
  
```



```

JBL=HW2
HW4=FTRA
X(CTRA+1)=FW2
FTRA=CTRA
TRMCNT=TRMCNT-OS(1)
TCNT=TRMCNT
ZRTN=2108
GO TO 232
IF (TCNT.LE.0) GO TO 9999
2108      GOTO 400
C 211      GOTO (2111,2112,2113,2114,2115,2116),MDFR
C 2111     IF(OS(1).EQ.OS(2)) GOTO 230
C 2112     IF(OS(1).LT.OS(2)) GOTO 230
C 2113     IF(OS(1).LE.OS(2)) GOTO 230
C 2114     IF(OS(1).GT.OS(2)) GOTO 230
C 2115     IF(OS(1).GE.OS(2)) GOTO 230
C 2116     IF(OS(1).NE.OS(2)) GOTO 230
2117     ALTBLO=OS(3)
2117     IF(ALTBLQ.EQ.0) GOTO 4002
2117     NEXBLO=ALTBLQ
2117     GOTO 231
C 212      TABULATE
212      IF(OS(2).GT.NTAB) CALL ERRORZ(435,0,&9999)
212      RARG=OS(1)
212      IF(INTDEC(1).EQ.1) RARG=FOS(1)
212      TBLNO=OS(2)
212      WTFAC=OS(3)
212      IF(INTDEC(3).EQ.1) WTFAC=FOS(3)
212      IF(WTFAC.EQ.0) WTFAC=1
212      N=X(TAB+TBLNO)
212      RX(N+1)=RX(N+1)+RARG
212      RX(N+2)=RX(N+2)+((RARG**2)
212      RX(N+3)=RX(N+3)+((RARG*WTFAC)
212      RX(N+4)=RX(N+4)+((RARG**2)*WTFAC
212      X(N+5)=X(N+5)+WTFAC
212      FREWDT=X(N+6)

```

SIM08094
SIM08100
SIM08110
SIM08120
SIM08130
SIM08132
SIM08134
SIM08136
SIM08140
SIM08160

SIM08180
SIM08200
SIM08210
SIM08230
SIM08240
SIM08260
SIM08270
SIM08290
SIM08300
SIM08320
SIM08330
SIM08350
SIM08360
SIM08370
SIM08380
SIM08390

SIM08410
SIM08420
SIM08430
SIM08440
SIM08450
SIM08460
SIM08470
SIM08480
SIM08490
SIM08500
SIM08510
SIM08520
SIM08530
SIM08540

MRNG=X(N+8)				SIM08550
ULLI=X(N+9)				SIM08560
PNTF=N+9				SIM08570
J=(RARG-ULLI)/FREWDI+2.				SIM08580
IF (J.LT.1) J=1				SIM08590
IF (J.GT.MRNG) J=MRNG				SIM08600
IF (J.PNTF+J)=X(PNTF+J)+WTFAC				SIM08610
2122 X(PNTF+J)=X(PNTF+J)+WTFAC				SIM08620
IF (J.EQ.MRNG) RX(N+7)=RX(N+7)+RARG				SIM08630
GOTO 230				
			SAVEVALUE	
C 213 IF(OS(1).GT.NSAV) CALL ERRORZ(433,0,&9999)				SIM08650
IF(MDFR-2) 2131,2132,2133				SIM08660
2131 X(SAVE+OS(1))=X(SAVE+OS(1))-OS(2)				SIM08670
GOTO 230				SIM08680
2132 X(SAVE+OS(1))=OS(2)				SIM08690
GOTO 230				SIM08700
2133 X(SAVE+OS(1))=X(SAVE+OS(1))+OS(2)				SIM08710
GOTO 230				SIM08720
			GATE	
C 216 GOTO (217,218,219,220),MDFR				SIM08740
			GATE SNE	
C 217				
CSTO=X(STO+OS(1))				SIM08760
ALTBLO=OS(2)				SIM08770
USED=X(CSTO+1)				SIM08780
IF(USED.NE.0) GOTO 230				SIM08790
IF(ALTBLQ.EQ.0) GOTO 2171				SIM08800
NEXBLO=ALTBLQ				SIM08810
GOTO 231				SIM08820
2171 FW1=X(CSTO+8)				SIM08830
FW2=X(CTRA+1)				SIM08840
HW3=HW2				SIM08850
HW2=CTRA				SIM08860
X(CSTO+8)=FW1				SIM08870
X(CTRA+1)=FW2				SIM08880
FW1=X(CTRA+6)				SIM08890
BYTE(2)=.FALSE.				SIM08900
X(CTRA+6)=FW1				SIM08910
GOTO 4002				SIM08920
			GATE SF	
C 218				
CSTO=X(STO+OS(1))				SIM08940
ALTBLO=OS(2)				SIM08950
AVAIL=X(CSTO+2)				SIM08960
IF(AVAIL.EQ.0) GOTO 230				SIM08970
IF(ALTBLQ.EQ.0) GOTO 2181				SIM08980
NEXBLO=ALTBLQ				SIM08990
GOTO 231				SIM09000
2181 FW1=X(CSTO+8)				SIM09010
FW2=X(CTRA+1)				SIM09020

SIM09030
SIM09040
SIM09050

GATE SNF

SIM09070
SIM09080
SIM09090
SIM09100
SIM09110
SIM09120
SIM09130
SIM09140
SIM09150
SIM09160
SIM09170
SIM09180

GATE SE

SIM09200
SIM09210
SIM09220
SIM09230
SIM09240
SIM09250
SIM09260
SIM09270
SIM09280
SIM09290
SIM09300
SIM09310

SELECT

SIM09330
SIM09340
SIM09350
SIM09360
SIM09370
SIM09380
SIM09390
SIM09400
SIM09410
SIM09420
SIM09430
SIM09440
SIM09450
SIM09460
SIM09470
SIM09480
SIM09490
SIM09500

HW3=HW1
HW1=CTRA
GOTO 2172

C 219 CSTO=X(STO+OS(1))
ALTBLO=OS(2)
AVAIL=X(CSTO+2)
IF(AVAIL.NE.0) GOTO 230
IF(ALTBL0.EQ.0) GOTO 2191
NEXBLO=ALTBL0
GOTO 231

2191

FW1=X(CSTO+9)
FW2=X(CTRA+1)
HW3=HW2
HW2=CTRA
GOTO 2172

C

C 220 CSTO=X(STO+OS(1))
ALTBLO=OS(2)
USED=X(CSTC+1)
IF(USED.EQ.0) GOTO 230
IF(ALTBL0.EQ.0) GOTO 2201
NEXBLO=ALTBL0
GOTO 231

2201

FW1=X(CSTO+9)
FW2=X(CTRA+1)
HW3=HW1
HW1=CTRA
GOTO 2172

C

C 221 TPAR=OS(1)
LOW=OS(2)
HIGH=OS(3)
COMVAL=OS(4)
ALTBLO=OS(6)
IF(INTDEC(4).EQ.1) COMVAL=FOS(4)
TRTN=ZRTN
ZRTN=2211
TEMP1=X(CBLO+1)
TEMP2=X(CBLO+2)
DO 2230 I=LOW, HIGH
HW1=-33
HW2=0
X(CBLO+2)=FW1
FW1=X(CBLO+6)
HW2=I
X(CBLO+1)=FW1
INST=CBLO+1

2211	GOTO 100	SIM09510
	J=0	SIM09520
	GOTO (2213,2214,2215,2216,2217,2218,2219,2220,	SIM09530
	2221,2222,2223,2224),MDER	SIM09540
2213	- IF(COMVAL.EQ.OS(1)) GOTO 2231	SIM09550
	GOTO 2230	SIM09560
2214	IF(COMVAL.LT.OS(1)) GOTO 2231	SIM09570
	GOTO 2230	SIM09580
2215	IF(COMVAL.LE.OS(1)) GOTO 2231	SIM09590
	GOTO 2230	SIM09600
2216	IF(COMVAL.GT.OS(1)) GOTO 2231	SIM09610
	GOTO 2230	SIM09620
2217	IF(COMVAL.GE.OS(1)) GOTO 2231	SIM09630
	GOTO 2230	SIM09640
2218	IF(COMVAL.NE.OS(1)) GOTO 2231	SIM09650
	GOTO 2230	SIM09660
2219	IF(I.EQ.HIGH) GOTO 2231	SIM09670
	IF(J.GE.OS(1)) GOTO 2231	SIM09680
	J=OS(1)	SIM09690
	K=I	SIM09700
	GOTO 2230	SIM09710
2220	IF(I.EQ.HIGH) GOTO 2231	SIM09720
	IF(J.LE.OS(1)) GOTO 2231	SIM09730
	J=OS(1)	SIM09740
	K=I	SIM09750
	GOTO 2230	SIM09760
2221	IF(OS(1).GT.0) GOTO 2231	SIM09770
	GOTO 2230	SIM09780
2222	IF(OS(1).EQ.0) GOTO 2231	SIM09790
	GOTO 2230	SIM09800
2223	IF(OS(1).NE.0) GOTO 2231	SIM09810
	GOTO 2230	SIM09820
2224	IF(OS(1).EQ.0) GOTO 2231	SIM09830
	GOTO 2230	SIM09840
2230	CONTINUE	SIM09850
2231	X(CBLO+1)=TEMP1	SIM09860
	X(CBLO+2)=TEMP2	SIM09870
	X(CTRA+8+TPAR)=J	SIM09880
	IF(ALTBL0.EQ.0) GOTO 230	SIM09890
	NEXBLO=ALTBL0	SIM09900
	GOTO 231	SIM09910
229	CALL ERRORZ(603,11,£9999)	SIM09920
C		
230	FW1=X(CTRA+5)	SIM09940
	FW2=X(CTRA+1)	SIM09950
	JBL=HW4	SIM09960
	HW4=HW1	SIM09970
	HW1=HW1+1	SIM09980

RETURN FROM BLOCK ROUTINES

SIM09990
SIM10000
SIM10010
SIM10020
SIM10030
SIM10040
SIM10050
SIM10060
SIM10070
SIM10080
SIM10090
SIM10100
SIM10110
SIM10120
SIM10130
SIM10140
SIM10150
SIM10160
SIM10170
SIM10180
SIM10190
SIM10200
SIM10210
SIM10220
SIM10230
SIM10240
SIM10250
SIM10260
SIM10270
SIM10280
SIM10290
SIM10300

SIM10330
SIM10340
SIM10350
SIM10360
SIM10370
SIM10380
SIM10390
SIM10400
SIM10410
SIM10420
SIM10430
SIM10440

```

231  GOTO 2311
      FW1=X(CTRA+5)
      FW2=X(CTRA+1)
      JBL=HW4
      HW4=HW1
      HW1=NEXBLO
2311  X(CTRA+5)=FW1
232  X(CTRA+1)=FW2
      N=X(BLOCK+JBL)
      FW1=X(N+1)
      IF(CTRBLO) WRITE(OUT6,9040) HW1
      IF(HW1.EQ.1) GOTO 2313
      IF(.NOT.CTRBLO) GOTO 2315
2312  WRITE(OUT6,9031)
      WRITE(OUT6,9032) CTRA,FTCEC,LTCEC,FTFEC,LTCEC
2315  IF(ZRTN.EQ.4005) GOTO 4005
      IF(ZRTN.EQ.2108) GO TO 2108
      CALL ERRORZ(603,12,&9999)
2313  WRTN=ZRTN
      TTRA=CTRA
      TBLO=CBLO
      ZRTN=2316
      INST=N+2
      GOTO 100
2316  OP CODE=1
      CBLO=X(BLOCK+JBL)
      ZRTN=2314
      GOTO 200
2314  CTRA=TTRA
      CBLO=TBLO
      ZRTN=WRTN
      GOTO 2312
299  CALL ERRORZ(2,0,&9999)

      *** UPDATE CLOCK MOVE TRANS
      FROM FEC TO CEC ***

C
C
300  IF(FTFEC.EQ.0) CALL ERRORZ(401,0,&9999)
      CLOCK=X(FTFEC+3)
      IF(TRUPD) WRITE(OUT6,9023) CLOCK
3001  IF(NOUPD.EQ.1) GOTO 3002
      CKPNT=1
      IF(TRS1) WRITE(OUT6,9042)
      IF(TRS1) WRITE(OUT6,9043) CKPNT,FTFEC,LTCEC,FTCEC,LTCEC,
      -CTRA,PNTT,FTRA,X(CTRA+1),X(CTRA+2),X(CTRA+3),X(PNTT+1),X(PNTT+2),
      -X(PNTT+3),NEWBDT,OLDBDT
      CTRA=FTFEC
      FW1=X(CTRA+1)
      FTFEC=HW2

```



```

IF(FTFEC.EQ.0) GOTO 3002
FW1=X(FTFEC+1)
HW1=0
X(CTRA+1)=FW1
3002 CKPNT=2
IF(TRS1) WRITE(OUT6,9042)
IF(TRS1) WRITE(OUT6,9043) CKPNT,FTFEC,LTCEC,FTCEC,LTCEC,
-CTRA,PNTT,FTRA,X(CTRA+1),X(PNTT+1),X(PNTT+2),
-X(PNTT+3),NEWBDT,OLDBDT
IF(FTCEC.LE.0) GOTO 3003
FW1=X(CTRA+5)
NEWPRI=HW2
HW2=FTCEC
GOTO 3004
3003 FTCEC=CTRA
CKPNT=3
IF(TRS1) WRITE(OUT6,9042)
IF(TRS1) WRITE(OUT6,9043) CKPNT,FTFEC,LTCEC,FTCEC,LTCEC,
-CTRA,PNTT,FTRA,X(CTRA+1),X(PNTT+1),X(PNTT+2),
-X(PNTT+3),NEWBDT,OLDBDT
LTCEC=FTCEC
X(CTRA+2)=0
GOTO 3005
3004 CKPNT=4
IF(TRS1) WRITE(OUT6,9042)
IF(TRS1) WRITE(OUT6,9043) CKPNT,FTFEC,LTCEC,FTCEC,LTCEC,
-CTRA,PNTT,FTRA,X(CTRA+1),X(PNTT+1),X(PNTT+2),
-X(PNTT+3),NEWBDT,OLDBDT
FW2=X(PNTT+5)
OLDPRI=HW4
FW1=X(PNTT+2) GOTO 3006
IF(HW2.EQ.0) GOTO 3006
IF(NEWPRI.LE.OLDPRI) GOTO 3004
FW2=X(CTRA+2)
HW4=HW2
HW3=PNTT
X(CTRA+2)=FW2
FW2=X(HW2+2)
HW3=CTRA
X(HW2+2)=FW2
HW2=CTRA
X(PNTT+2)=FW1
IF(TRUPD) WRITE(OUT6,9033)
IF(TRUPD) WRITE(OUT6,9032) PNTT,CTRA,FTCEC,LTCEC,FTFEC,LTCEC
3005 IF(NUUPD.EQ.1) GOTO 2014
IF(FTFEC.LE.0) GOTO 400

```

```

SIM10450
SIM10460
SIM10470
SIM10480
SIM10490
SIM10500
SIM10510
SIM10520
SIM10530
SIM10540
SIM10550
SIM10560
SIM10570
SIM10580
SIM10590
SIM10600
SIM10610
SIM10620
SIM10630
SIM10640
SIM10650
SIM10660
SIM10670
SIM10680
SIM10690
SIM10700
SIM10710
SIM10720
SIM10730
SIM10740
SIM10750
SIM10760
SIM10770
SIM10780
SIM10790
SIM10800
SIM10810
SIM10820
SIM10830
SIM10840
SIM10850
SIM10860
SIM10870
SIM10880
SIM10890
SIM10900
SIM10910
SIM10920

```



```

CKPNT= 5      WRITE(OUT6,9042)
IF( (TRSI) )  WRITE(OUT6,9043)  CKPNT,FTFEC,LTSEC,LTCEC,
-CTRA,PNTT,FTRA,X(CTRA+1),X(CTRA+2),X(CTRA+3),X(PNTT+1),X(PNTT+2),
-X(PNTT+3),NEWBDT,OLDBDT
NEXBDT=X(FTFEC+3)
IF(NEXBDT.GT.CLOCK) GOTO 400
GOTO 3001
3006 IF(NEWPRI.GT.OLDPRI) GOTO 3007
CKPNT= 6      WRITE(OUT6,9042)
IF( (TRSI) )  WRITE(OUT6,9043)  CKPNT,FTFEC,LTSEC,LTCEC,
-CTRA,PNTT,FTRA,X(CTRA+1),X(CTRA+2),X(CTRA+3),X(PNTT+1),X(PNTT+2),
-X(PNTT+3),NEWBDT,OLDBDT
FW2=X(PNTT+2)
HW4=CTRA
X(PNTT+2)=FW2
FW2=X(CTRA+2)
HW3=PNTT
HW4=0
X(CTRA+2)=FW2
LTCEC=CTRA
IF( (NOUPD.EQ.1) ) GOTO 2014
GOTO 3005
3007 FW2=X(CTRA+2)
CKPNT= 7      WRITE(OUT6,9042)
IF( (TRSI) )  WRITE(OUT6,9043)  CKPNT,FTFEC,LTSEC,LTCEC,
-CTRA,PNTT,FTRA,X(CTRA+1),X(CTRA+2),X(CTRA+3),X(PNTT+1),X(PNTT+2),
-X(PNTT+3),NEWBDT,OLDBDT
HW3=HW1
HW4=PNTT
X(CTRA+2)=FW2
HW1=CTRA
X(PNTT+2)=FW1
IF( (NOUPD.EQ.1) ) GOTO 2014
GOTO 3005

*** UNLINK FROM CEC AND
MERGE INTO FEC ***

C 310 FW1=X(CTRA+2)
CTRA=HW2
CKPNT= 8      WRITE(OUT6,9042)
IF( (TRSI) )  WRITE(OUT6,9043)  CKPNT,FTFEC,LTSEC,LTCEC,
-CTRA,PNTT,FTRA,X(CTRA+1),X(CTRA+2),X(CTRA+3),X(PNTT+1),X(PNTT+2),
-X(PNTT+3),NEWBDT,OLDBDT
IF( (HW1.EQ.0) ) GOTO 3101

```

```

SIM10930
SIM10940
SIM10950
SIM10960
SIM10970
SIM10980
SIM10990
SIM11000
SIM11010
SIM11020
SIM11030
SIM11040
SIM11050
SIM11060
SIM11070
SIM11080
SIM11090
SIM11100
SIM11110
SIM11120
SIM11130
SIM11140
SIM11150
SIM11160
SIM11170
SIM11180
SIM11190
SIM11200
SIM11210
SIM11220
SIM11230
SIM11240
SIM11250
SIM11260
SIM11270
SIM11280
SIM11290

SIM11320
SIM11330
SIM11340
SIM11350
SIM11360
SIM11370
SIM11380
SIM11390

```



```

FW2=X(HW1+2)
HW4=HW2
X(HW1+2)=FW2
GOTO 3102
FTCEC=HW2
3101 CKPNT=9
IF(TRS1) WRITE(OUT6,9042)
IF(TRS1) WRITE(OUT6,9043) CKPNT,FTFEC,LTSEC,FTCEC,LTCEC,
-CTRA,PNTT,FTRA,X(CTRA+1),X(CTRA+2),X(CTRA+3),X(PNTT+1),X(PNTT+2),
-X(PNTT+3),NEWBDT,OLCDBT
IF(HW2.EQ.0) GOTO 3103
3102 CKPNT=10
IF(TRS1) WRITE(OUT6,9042)
IF(TRS1) WRITE(OUT6,9043) CKPNT,FTFEC,LTSEC,FTCEC,LTCEC,
-CTRA,PNTT,FTRA,X(CTRA+1),X(CTRA+2),X(CTRA+3),X(PNTT+1),X(PNTT+2),
-X(PNTT+3),NEWBDT,OLCDBT
FW2=X(HW2+2)
HW3=HW1
X(HW2+2)=FW2
GOTO 3104
3103 LTCEC=HW1
CKPNT=11
IF(TRS1) WRITE(OUT6,9042)
IF(TRS1) WRITE(OUT6,9043) CKPNT,FTFEC,LTSEC,FTCEC,LTCEC,
-CTRA,PNTT,FTRA,X(CTRA+1),X(CTRA+2),X(CTRA+3),X(PNTT+1),X(PNTT+2),
-X(PNTT+3),NEWBDT,OLCDBT
UNLINKED
C 3104 X(CTRA+2)=X(CTRA+1)
CKPNT=12
IF(TRS1) WRITE(OUT6,9042)
IF(TRS1) WRITE(OUT6,9043) CKPNT,FTFEC,LTSEC,FTCEC,LTCEC,
-CTRA,PNTT,FTRA,X(CTRA+1),X(CTRA+2),X(CTRA+3),X(PNTT+1),X(PNTT+2),
-X(PNTT+3),NEWBDT,OLCDBT
IF(FTFEC.EQ.0) GOTO 3108
NEWBDT=X(CTRA+3)
HW2=FTFEC
PNTT=HW2
3105 CKPNT=13
IF(TRS1) WRITE(OUT6,9042)
IF(TRS1) WRITE(OUT6,9043) CKPNT,FTFEC,LTSEC,FTCEC,LTCEC,
-CTRA,PNTT,FTRA,X(CTRA+1),X(CTRA+2),X(CTRA+3),X(PNTT+1),X(PNTT+2),
-X(PNTT+3),NEWBDT,OLCDBT
OLCDBT=X(PNTT+3)
FW1=X(PNTT+1)
IF(HW2.LE.0) GOTO 3107
IF(NEWBDT.GE.OLCDBT) GOTO 3105
IF(HW1.EQ.0) FTFEC=CTRA
HW3=HW1

```

```

SIM11400
SIM11410
SIM11420
SIM11430
SIM11440
SIM11450
SIM11460
SIM11470
SIM11480
SIM11490
SIM11500
SIM11510
SIM11520
SIM11530
SIM11540
SIM11550
SIM11560
SIM11570
SIM11580
SIM11590
SIM11600
SIM11610
SIM11620
SIM11630
SIM11640
SIM11650
SIM11670
SIM11680
SIM11690
SIM11700
SIM11710
SIM11720
SIM11730
SIM11740
SIM11750
SIM11760
SIM11770
SIM11780
SIM11790
SIM11800
SIM11810
SIM11820
SIM11830
SIM11840
SIM11850
SIM11860
SIM11870

```



```

HW4=PNTT
X(CTRA+1)=FW2
IF(HW3.EQ.0) GOTO 3111
FW3=X(HW3+1)
HW6=CTRA
X(HW3+1)=FW3
GOTO 3112
3106 HW4=PNTT
CKPNT=14
WRITE(OUT6,9042)
IF(TRS1) WRITE(OUT6,9043) CKPNT,FTFEC,LTSEC,FTCEC,LTCEC,
-CTRA,PNTT,FTRA,X(CTRA+1),X(PNTT+1),X(PNTT+2),
-X(PNTT+3),NEWBDT,OLDBDT
HW3=HW1
X(CTRA+1)=FW2
IF(HW1.EQ.0) GOTO 3111
FW2=X(HW1+1)
CKPNT=15
WRITE(OUT6,9042)
IF(TRS1) WRITE(OUT6,9043) CKPNT,FTFEC,LTSEC,FTCEC,LTCEC,
-CTRA,PNTT,FTRA,X(CTRA+1),X(PNTT+1),X(PNTT+2),
-X(PNTT+3),NEWBDT,OLDBDT
HW4=CTRA
X(HW1+1)=FW2
GOTO 3112
3111 FTFEC=CTRA
CKPNT=16
WRITE(OUT6,9042)
IF(TRS1) WRITE(OUT6,9043) CKPNT,FTFEC,LTSEC,FTCEC,LTCEC,
-CTRA,PNTT,FTRA,X(CTRA+1),X(PNTT+1),X(PNTT+2),
-X(PNTT+3),NEWBDT,OLDBDT
HW1=CTRA
CKPNT=17
WRITE(OUT6,9042)
IF(TRS1) WRITE(OUT6,9043) CKPNT,FTFEC,LTSEC,FTCEC,LTCEC,
-CTRA,PNTT,FTRA,X(CTRA+1),X(PNTT+1),X(PNTT+2),
-X(PNTT+3),NEWBDT,OLDBDT
X(PNTT+1)=FW1
GOTO 3109
3107 IF(NEWBDT.LT.OLDBDT) GOTO 3106
CKPNT=18
WRITE(OUT6,9042)
IF(TRS1) WRITE(OUT6,9043) CKPNT,FTFEC,LTSEC,FTCEC,LTCEC,
-CTRA,PNTT,FTRA,X(CTRA+1),X(PNTT+1),X(PNTT+2),
-X(PNTT+3),NEWBDT,OLDBDT
LTFEC=CTRA
HW4=0
HW3=PNTT

```

SIM11880
SIM11890
SIM11900
SIM11910
SIM11920
SIM11930
SIM11940
SIM11950
SIM11960
SIM11970
SIM11980
SIM11990
SIM12000
SIM12010
SIM12020
SIM12030
SIM12040
SIM12050
SIM12060
SIM12070
SIM12080
SIM12090
SIM12100
SIM12110
SIM12120
SIM12130
SIM12140
SIM12150
SIM12160
SIM12170
SIM12180
SIM12190
SIM12200
SIM12210
SIM12220
SIM12230
SIM12240
SIM12250
SIM12260
SIM12270
SIM12280
SIM12290
SIM12300
SIM12310
SIM12320
SIM12330
SIM12340
SIM12350

SIM12360
SIM12370
SIM12380
SIM12390
SIM12400
SIM12410
SIM12420
SIM12430
SIM12440
SIM12450
SIM12460
SIM12470
SIM12480
SIM12490
SIM12500
SIM12510
SIM12520
SIM12530

```

X(CTRA+1)=FW2
HW2=CTRA
X(PNTT+1)=FW1
GOTO 3109
FTFEC=CTRA
CKPNT=19
IF(TRS1) WRITE(OUT6,9042)
IF(TRS1) WRITE(OUT6,9043) CKPNT,FTFEC,LTFEC,FTCEC,LTFEC,X(PNTT+1),X(PNTT+2),
-CTRA,PNTT,FTRA,X(CTRA+1),X(CTRA+2),X(CTRA+3),X(PNTT+1),X(PNTT+2),
-X(PNTT+3),NEWBDT,OLDBDT
LTFEC=FTFEC
X(CTRA+1)=0
IF(ZRTN.EQ.30) GOTO 30
IF(ZRTN.EQ.2314) GOTO 2314
CTRA=CTRA
IF(CTRA.EQ.0) GOTO 400
IF(ZRTN.EQ.400) GOTO 400
CALL ERRCRZ(603,13,89999)

```

*** SCAN ***

SIM12550
SIM12560
SIM12570
SIM12580
SIM12590
SIM12600
SIM12610

```

C 400 IF(TRSCN) WRITE(OUT6,9024)
SCFLAG=0
CTRA=FTCEC
IF(CTRA.EQ.0) GOTO 300
FW1=X(CTRA+6)
SSIND=BYTE(2)
IF(SSIND) GOTO 4003
IF(SCFLAG.EQ.1) GOTO 400
FW1=X(CTRA+2)
IF(HW2.EQ.0) GOTO 300
CTRA=HW2
GOTO 4008
FW1=X(CTRA+5)
CBLO=X(BLOCK+HW1)
INST=CBLO+2
FW1=X(CBLO+1)
OPCODE=HW1
ZRTN=4004
GOTO 100
ZRTN=4005
GOTO 200
FW1=X(CBLO+1)
OPCODE=HW1
IF(TRSCN) WRITE(OUT6,9034)
IF(TRSCN) WRITE(OUT6,9035) SCFLAG,CBLO,OPCODE,INST
IF(OPCODE.NE.2) GOTO 4003
NEWBDT=X(CTRA+3)
IF(NEWBDT.LE.CLOCK) GOTO 4003

```

SIM12620
SIM12630
SIM12640
SIM12650
SIM12660
SIM12670
SIM12680
SIM12690
SIM12700
SIM12710
SIM12720
SIM12730
SIM12740
SIM12750
SIM12760
SIM12770
SIM12780
SIM12800
SIM12810
SIM12820

```

4003
4004
4005
4006

```



```

IF (TRSCN) WRITE (OUT6,9030) NEWBDT
ZRTN=400
GOTO 310

```

C
C
C

*** UPDATE FINAL INTERVAL ***

```

9999 IF (ERR.NE.0) GO TO 5900
      IF (NSTO.EQ.0) GO TO 5200
      DO 5150 I=1,NSTO
      N=X(STO+I)
      IF (N.EQ.0.OR.X(N+6).EQ.0) GO TO 5150
      DTIME=CLOCK-X(N+5)
      X(N+5)=CLOCK
      FW1=X(N+3)
      FW2=X(N+4)
      DW1=DW1+DTIME*X(N+1)
      X(N+3)=FW1
      X(N+4)=FW2
      CONTINUE
5150 IF (NQUE.EQ.0) GO TO 5300
5200 IF (NQUE.I=1,NQUE
      N=X(QUE+I)
      IF (N.EQ.0.OR.X(N+2).EQ.0) GO TO 5250
      DTIME=CLOCK-X(N+1)
      X(N+1)=CLOCK
      FW1=X(N+4)
      FW2=X(N+5)
      DW1=DW1+DTIME*X(N+6)
      X(N+4)=FW1
      X(N+5)=FW2
      CONTINUE
5250 CLOCK=CLOCK-CLOCKB
5300 WRITE (OUT6,9550) CLOCKR,CLOCK
      GO TO 5910
5900 WRITE (OUT6,9026)
5910 IF (OUTX.EQ.0) RETURN
      WRITE (OUTX,9020)
      WRITE (OUTX,9018)
      DO 5930 I=1,TRSIZE
      FW1=X(I)
      IF (FW1.GE.16000.OR.HW1.LT.-16000) GO TO 5920
      WRITE (OUTX,9019) I,FW1,HW1,HW2
      GO TO 5930
5920 WRITE (OUTX,9027) I,FW1,HW1,HW2,FW1
5930 CONTINUE
      RETURN

```

SIM12830
SIM12840
SIM12850
SIM12860
SIM12870
SIM12880

SIM12900
SIM12910
SIM12920
SIM12930
SIM12940
SIM12950
SIM12960
SIM12970
SIM12980
SIM12990
SIM13000
SIM13010
SIM13020
SIM13030
SIM13040
SIM13050
SIM13060
SIM13070
SIM13080
SIM13090
SIM13100
SIM13110
SIM13120
SIM13130
SIM13140
SIM13150
SIM13160
SIM13170
SIM13180
SIM13190
SIM13200
SIM13210
SIM13220
SIM13230
SIM13240
SIM13250
SIM13260
SIM13270
SIM13280
SIM13285

***** SPSTAT *****

SIM13320
SIM13330
SIM13340
SIM13350
SIM13360
SIM13365
SIM13370
SIM13380
SIM13390
SIM13400
SIM13410
SIM13420
SIM13430
SIM13440
SIM13450
SIM13470
SIM13480
SIM13500
SIM13510

```

ENTRY SPSTAT(SNA,IDT,VAL,IORD)
INTEGER*2 SNA,IDT,IORD
HW1=SNA
HW2=IDT
VAL=0
IF (CLOCK.EQ.0) RETURN
RVAL=0.
EPT=2
IF (SNA.GE.5.AND.SNA.LE.21.OR.SNA.GE.27.AND.SNA.LE.29) GO TO 1006
WRITE (6,9521)
RETURN

```

```

C 5950 IF (RVAL.EQ.0.) GO TO 5952
      IORD=1
      FFW1=RVAL
      GO TO 5954
5952 IORD=0
5954 VAL=FW1
      RETURN

```

***** SIMOUT *****

SIM13550
SIM13560
SIM13570
SIM13580
SIM13590
SIM13600
SIM13610
SIM13620
SIM13630
SIM13640
SIM13650
SIM13660
SIM13670
SIM13680
SIM13690
SIM13700

```

ENTRY SIMOUT(OUT6)
INTEGER*2 I2,HC1 /1/,HC55 /55/,HC2 /2/,HC3 /3/
REAL*8 STATSW(4),BITTS
      OUTPUT STORAGE STATISTICS
CALL GBITS(HC1,HC55,STATSW(1),BITTS)
IF (NSTO.EQ.0.OR.BITTS.EQ.0.) GO TO 6100
WRITE (OUT6,9500)
DO 6050 I2=1,NSTO
CALL GBITS(I2,I2,STATSW(1),BITTS)
N=X(STO+I2)
IF (N.EQ.0.OR.BITTS.EQ.0.) GO TO 6050
FW1=X(N+3)
FW2=X(N+4)
FW3=X(N+1)+X(N+2)
RR1=DW1/CLOCK
RR2=DW1/(CLOCK*FW3)
RR3=0.

```



```

        IF (X(N+6).NE.0) RR3=DW1/X(N+6)
        WRITE (OUT6,9101) I2,FW3,RR1,RR2,X(N+6),RR3,X(N+1),X(N+7)
        CONTINUE
C
6050
C
6100
        CALL GBITS(HC1,HC55,STATSW(2),BITTS)
        IF (NQUE.EQ.0.OR.BITTS.EQ.0.) GO TO 6200
        WRITE (OUT6,9501)
        DO 6150 I2=1,NQUE
        CALL GBITS(I2,I2,STATSW(2),BITTS)
        N=X(QUE+I2)
        IF (N.EQ.0.OR.BITTS.EQ.0.) GO TO 6150
        FW1=X(N+4)
        FW2=X(N+5)
        RR1=DW1/CLOCK
        RR2=0.
        IF (X(N+2).NE.0) RR2=100.*X(N+3)/X(N+2)
        RR3=0.
        IF (X(N+2).NE.0) RR3=DW1/X(N+2)
        RR4=0.
        IF (X(N+2).NE.X(N+3)) RR4=DW1/(X(N+2)-X(N+3))
        WRITE (OUT6,9103) I2,X(N+7),RR1,X(N+2),X(N+3),RR2,RR3,RR4,X(N+6)
        CONTINUE
        WRITE (OUT6,9502)
C
6150
C
6200
        CALL GBITS(HC1,HC55,STATSW(3),BITTS)
        IF (NTAB.EQ.0.OR.BITTS.EQ.0.) GO TO 6300
        DO 6250 I2=1,NTAB
        CALL GBITS(I2,I2,STATSW(3),BITTS)
        N=X(TAB+I2)
        IF (N.EQ.0.OR.BITTS.EQ.0.) GO TO 6250
        IF (X(N+5).GT.1) GO TO 6210
        RR1=RX(N+1)
        RR2=0.
        GO TO 6220
        RR1=RX(N+1)/X(N+5)
        RR2=SQRT((RX(N+2)-RX(N+1)**2/X(N+5))/(X(N+5)-1))
        WRITE (OUT6,9503) I2,X(N+5),RR1,RR2,RX(N+1)
        IF (X(N+5).EQ.0) GO TO 6245
        FRNG=X(N+8)
        RR4=0.
        DO 6240 J=1,FRNG
        FW1=X(N+9+J)
        RR3=100.*FW1/X(N+5)
        RR4=RR4+RR3
        RR5=100.-RR4
        FW2=X(N+9)+(J-1)*X(N+6)
        IF (RR1.EQ.0.) GO TO 6225
        RR6=FW2/RR1

```

SIM13710
SIM13720
SIM13730

SIM13740
SIM13750
SIM13760
SIM13770
SIM13780
SIM13790
SIM13800
SIM13810
SIM13820
SIM13830
SIM13840
SIM13850
SIM13860
SIM13870
SIM13875
SIM13880
SIM13890
SIM13900
SIM13910

SIM13920
SIM13930
SIM13940
SIM13950
SIM13960
SIM13970
SIM13980
SIM13990
SIM14000
SIM14010
SIM14020
SIM14030
SIM14040
SIM14050
SIM14060
SIM14070
SIM14080
SIM14090
SIM14100
SIM14110
SIM14120
SIM14130
SIM14140
SIM14150

6225	GO TO 6228	SIM14160
6228	RR6=0.	SIM14170
	IF (RR2.EQ.0.) GO TO 6230	SIM14180
	RR7=(FW2-RR1)/RR2	SIM14190
	GO TO 6232	SIM14200
6230	RR7=0.	SIM14210
6232	IF (J.EQ.FRNG) GO TO 6235	SIM14220
	WRITE(OUT6,9110) FW2,FW1,RR3,RR4,RR5,RR6,RR7	SIM14230
	GO TO 6240	SIM14240
6235	IF (FW1.EQ.0) GO TO 6245	SIM14250
	FW3=RX(N+7)/FW1	SIM14260
	WRITE(OUT6,9509) FW1,RR3,RR4,RR5,FFW3	SIM14270
6240	CONTINUE	SIM14280
	GO TO 6250	SIM14290
6245	WRITE(OUT6,9510)	SIM14300
6250	CONTINUE	SIM14310
6300	CALL GBITS(HC1,HC1,STATSW(4),BITTS)	SIM14320
	IF (NSAV.EQ.0.OR.BITTS.EQ.0.) GO TO 6400	SIM14330
	WRITE(OUT6,9504)	SIM14340
	DO 6350, I=1,NSAV	SIM14350
	IF (X(SAVE+I).NE.0) WRITE(OUT6,9112) I,X(SAVE+I)	SIM14360
6350	CONTINUE	SIM14370
6400	FLAG=0	SIM14380
	CALL GBITS(HC2,HC2,STATSW(4),BITTS)	SIM14390
	IF (BITTS.EQ.0.) GO TO 6500	SIM14400
	N=FTCEC	SIM14410
	WRITE(OUT6,9505)	SIM14420
	WRITE(OUT6,9506)	SIM14430
6420	IF (N.EQ.0) GO TO 6500	SIM14440
	FW1=X(N+6)	SIM14450
	FW2=X(N+8)	SIM14460
	FW3=X(N+1)	SIM14470
	IF (FLAG.EQ.1) FW3=X(N+2)	SIM14480
	FW4=X(N+5)	SIM14490
	HW4=(N-TRAN)/(NPAR+8)+1	SIM14500
	WRITE(OUT6,9115) HW4,X(N+3),HW6,HW7,X(N+4),(X(N+8+J),J=1,4),	SIM14510
	-BYTE(2),BYTE(1)	SIM14520
	IF (NPAR.GT.4) WRITE(OUT6,9116) (X(N+8+J),J=5,NPAR)	SIM14530
	FW1=X(N+2-FLAG)	SIM14540
	N=HW2	SIM14550
	GO TO 6420	SIM14560
6500	IF (FLAG.NE.0) GO TO 6600	SIM14570
	CALL GBITS(HC3,HC3,STATSW(4),BITTS)	SIM14580
	IF (BITTS.EQ.0.) GO TO 6600	SIM14590
	FLAG=1	SIM14600
	N=FTFEC	SIM14610

SIM14620
SIM14630
SIM14640

SIM14650
SIM14660
SIM14670

SIM14680

ZERO OUTPUT STATISTIC
SWITCHES AND RETURN

```
C
C
6600 DO 6650 I=1,4
      STATSW(I)=0.
6650 CONTINUE
      WRITE (OUT6,9551)
      RETURN
```

***** SETIND *****

SIM14720
SIM14730
SIM14740

SET OUTPUT STATISTIC
SWITCHES AND RETURN

```
ENTRY SETIND(ROW,FB)
REAL*8 DONES /ZFFFFFFFFFFFFFFFFF/
INTEGER*2 FB,LB,ROW
```

SIM14745
SIM14750
SIM14760
SIM14770
SIM14780
SIM14790
SIM14800
SIM14810
SIM14820
SIM14830
SIM14840
SIM14850
SIM14860
SIM14870
SIM14880

```
IF (FB.LE.0) FB=60
IF (ROW.LT.1.OR.ROW.GT.6) ROW=6
IF (ROW.NE.6) GO TO 7100
DO 7050 I=1,4
  STATSW(I)=DONES
7050 RETURN
7100 I2=ROW
      IF (I2.EQ.5) I2=1
      LB=FB
      IF (FB.LE.55) GO TO 7130
      FB=1
      LB=55
7130 CALL PBITS(FB,LB,STATSW(I2),DONES)
      IF (ROW.EQ.5) CALL PBITS(FB,LB,STATSW(2),DONES)
      RETURN
```

***** FORMATS *****

SIM14940
SIM14950
SIM14960
SIM14970
SIM14980
SIM14990
SIM15000
SIM15010
SIM15020

```
9002 FORMAT(' SLCPE COMPUTATION')
9003 FORMAT(' BEGIN NEW SIMULATION')
9004 FORMAT(' CLEAR-CONTINUE SIMULATION')
9005 FORMAT(' RESET-CONTINUE SIMULATION')
9006 FORMAT(' EVALUATE ARGUMENTS')
9007 FORMAT(' PERFORM BLOCK ROUTINE',2X,'OPCODE=',I2,2X,'CBLD=',I5)
9015 FORMAT(I4,I4,I3,7I8)
9016 FORMAT(' ARG',I2,'=',F15.5)
9018 FORMAT(' 9X','HEX',8X,'DEC',6X,'DEC')
```



```

9505 FORMAT (////, 'CURRENT EVENTS CHAIN')
9506 17X, 'P1', 'TRANS', '5X', 'BDT', '4X', 'BLOCK', '4X', 'NBA', '4X', 'MARK--TIME',
9507 7X, 'P2', '7X', 'P3', '7X', 'P4', '4X', 'SI', '4X', 'TI', '/')
9508 17X, 'P1', 'FUTURE EVENTS CHAIN')
9509 7X, 'P2', 'SIMULATION TERMINATED')
9510 17X, 'P1', 'OVERFLOW', '5X', '17', '7X', 'F7.2, 6X, F8.2, 7X, F8.2,
9511 7X, 'P2', 'AVERAGE VALUE OF OVERFLOW', '2X, F12.3)
9512 17X, 'P1', 'REMAINING FREQUENCIES ARE ALL ZERO')
9513 7X, 'P2', 'INVALID SNA PASSED TO SPSTAT')
9514 17X, 'P1', 'SIMULATION TIME IS ', I10, ' (RELATIVE)', 'I10,
9515 7X, 'P2', ' (ABSOLUTE).')
9516 17X, 'P1', 'END
9517 7X, 'P2', 'END

100 SUBROUTINE ERRORZ(ERRNO, RECNO, *)
COMMON /XVECTR/ MAXX, X
INTEGER*4 ERRNO, RECNO, X(1000)
X(30)=ERRNO
WRITE(6,1001) ERRNO, RECNO
1001 FORMAT('OERRNO', 2X, I13, 2X, 'RECNO', 2X, I14)
RETURN 1
END
SIM15510
SIM15520
SIM15530
SIM15540
SIM15550
SIM15560
SIM15570
SIM15580
SIM15590
SIM15600
SIM15610
SIM15620
SIM15630
SIM15640
SIM15650

SIM15710
SIM15720
SIM15730
SIM15740
SIM15750
SIM15760
SIM15770
SIM15780

```


APPENDIX D

*** NLPQ RULE ADDITIONS AND MODIFICATIONS ***

```

NAMED RECORDS:
SETMEM      (-GPSSSW(MEM),-XVSW(MEM))
AVERAGE    (PS='ACJ',)
UTILIZA     (PS='NCUNS',INCGMP)
TRANSIT     (PS='NOUNS',)
CURRENT     (PS='NCUNS',)
MAXIMUM     (PS='ADJ',)
PRINT       (PS='ADJ',)
STORAGE     (PS='VERBS',NSFX)
QUEUE       (GPSSSENTY,PRCODE=1)
TABLE       (GPSSSENTY,PRCODE=2)
GPSSSENTY   (GPSSSENTY,PRCODE=3)
STATIST     (PS='NOUNS',)
GPSSATTR    (PS='NCUNS',INCOMP)
GPSSATTR    (PS='ATTR',)

SC           ('GPSSATTR',SNACODE=5,CHARS="NUMBER OF ENTRIES")
SM           ('GPSSATTR',SNACODE=6,CHARS="MAXIMUM CONTENTS")
SR           ('GPSSATTR',SNACODE=7,CHARS="MAXIMUM UTILIZATION")
SA           ('GPSSATTR',SNACODE=8,CHARS="AVERAGE CONTENTS")
S            ('GPSSATTR',SNACODE=9,CHARS="CURRENT CONTENTS")
R            ('GPSSATTR',SNACODE=10,CHARS="REMAINING SERVICE TIME")
ST           ('GPSSATTR',SNACODE=11,CHARS="AVERAGE LINE LENGTH")
Q            ('GPSSATTR',SNACODE=12,CHARS="CURRENT LINE LENGTH")
QA           ('GPSSATTR',SNACODE=13,CHARS="AVERAGE LINE LENGTH")
QM           ('GPSSATTR',SNACODE=14,CHARS="MAXIMUM LINE LENGTH")
QC           ('GPSSATTR',SNACODE=15,CHARS="MAXIMUM OF ENTRIES")
QZ           ('GPSSATTR',SNACODE=16,CHARS="NUMBER OF ZERO ENTRIES")
QT           ('GPSSATTR',SNACODE=17,CHARS="AVERAGE WAITING TIME")
QX           ('GPSSATTR',SNACODE=18,CHARS="AVERAGE WAITING TIME EXCLUDING ZERO ENTRIES")
TB           (CHARS="AVERAGE WAITING TIME",TIME EXCLUDING ZERO ENTRIES)
TC           ('GPSSATTR',SNACODE=19,CHARS="MEAN TRANSIT TIME")
TD           ('GPSSATTR',SNACODE=20,CHARS="NUMBER OF ENTRIES")
C            ('GPSSATTR',SNACODE=21,CHARS="STANDARD DEVIATION")
X            ('GPSSATTR',SNACODE=27,CHARS="CLOCK")
QP           ('GPSSATTR',SNACODE=28,CHARS="SAVEVALUE")
             ('GPSSATTR',SNACODE=29,CHARS="PERCENT ZERO ENTRIES")

ROUTINES:    SIMULT 21,    SIMOUT 22,    SETIND 23,    SPSTAT 24

```



```

DELETE XPARG E:
DELETE XCODE E:
DELETE XVECTOR E:

```

MORPHOLCGY FOR ENCODING X-VECTOR:

```

XPARG('$ACTION') -->
XVECTOR(RIGHT=IPDP(XPARG),LR('BACKSTUF')=LR('BACKSTUF')+1,
@LR('BACKSTUF')('BACKSTUF')=FX(MEM)+INDX(MEM),
LR('BACKSTUF')=LR('BACKSTUF')+1,
@LR('BACKSTUF')('BACKSTUF')=RIGHT)
XPARG('$ENTITY') -->
XVECTOR(RIGHT=ICNO(XPARG))
XPARG('$DECIMAL') -->
NULL(IX=FX(MEM)+INDX(MEM),XV=NUM(XPARG),
PUTFPX,-INDX(MEM),FX(MEM)=FX(MEM)+1)
XPARG('$ABSTIME') -->
TIMARG(%XPARG,XVSW)
XPARG('$QUANVAL') -->
XVECTOR(RIGHT=NUM(XPARG))
XPARG('$SNAREF,NUM') -->
XCODE(RIGHT=NUM(XPARG);NAM(XPARG),LEFT=22,
NAM.NE."FN",-LEFT)
XPARG('$SNAREF') -->
XCODE(RIGHT=IDNO(NAM2(XPARG)),NAM(XPARG),LEFT=12,
NAM.NE."Q",LEFT=13,NAM.NE."QA",-LEFT)
XPARG('$PARAMNO') -->
XCODE(LEFT=1,RIGHT=NUM(XPARG))
XPARG('$TRANSTM') -->
XCODE(LEFT=3,RIGHT=1)
XPARG('$RANDOM') -->
XCODE(LEFT=26,XRNNC(MEM)=XRNNC(MEM)+1,RIGHT=XRNNC(MEM),
XRNNC(MEM).EQ.8,-XRNNC(MEM))
XPARG('$NORMAL') -->
XCODE(LEFT=24,RIGHT=IDNO(XPARG))
XPARG(MEAN,STDEV) -->
XARG(DATA=STDEV(XPARG)) ...
XARG(DATA=MEAN(XPARG))
XCODE(LEFT=22,RIGHT=2) ...
XCODE(LEFT=3,RIGHT=3) ...
XCODE(LEFT=30,RIGHT=1) ...
XCODE(LEFT=25,RIGHT=0)
XPARG(XYLAST) -->
XCODE(LEFT=22,RIGHT=ICNO(XPARG))
XCODE -->
XVECTOR(%XCODE,LEFT=0-LEFT)
XVECTOR(INDX) -->
XVECTOR(INDX(MEM)=INDX,-INDX)
XVECTOR(-IX) -->
XVECTOR(IX=FX(MEM)+INDX(MEM),
-INDX(MEM),TEMP=0-LEFT,TEMP.NE.33,TEMP.NE.25,
FX(MEM)=FX(MEM)+1)
XVECTOR -->
NULL(%XVECTOR,XV=LEFT,PUTLX,XV=RIGHT,PUTRX)

```



```

DELETE QUEST2 E:
SEMOCLOGY FOR ENCODING ANSWERS TO QUESTIONS:
QUEST2(-ATTRIB) --> ANSWR('YES')
SENT(QUEST2,-PRD) -->
QUEST2(ATTRIB$,GPSSATTR) -->
SENT(QUEST2,@8=SNACODE(ATTRIB),@9=IDNQ,SPSTAT,@ATTRIB=@9)
QUEST2(-@ATTRIB) --> ANSWR('IDK')
QUEST2(-@VALIVAL$,INTRGVAL) --> QUEST5(%QUEST2)
QUEST2(-@ATTRIB,NE.ENTITY(VAL)) --> ANSWR('NC')
QUEST2(VALQUAN,-QUANTITY) --> QUEST5(%QUEST2)
QUEST2(VALQUAN,VALQUAN.NE.QUANTITY) --> ANSWR('IDK')
ANSWR('NO')
SENT(%QUEST2,ATTRIB='QUANTITY')
SENT(%QUEST2,'YES') QUEST5(%QUEST2)
QUEST2 -->

```

```

DELETE ENCODING E:
SEMOCLOGY FOR ENCODING - THE LINKAGE FROM DECODING:
ENCODING(ETYPE.EQ.'QUESTION') --> QUESTION(%ENCODING)
ENCODING(ETYPE.EQ.'REPLYERR') -->
PHRASE(CHARS=" THAT IS NOT A REASONABLE REPLY. TRY AGAIN.")
ENCODING(ETYPE(MEM).EQ.'ENGLISH') --> MASSAGER ENGLISH
ENCODING(ETYPE(MEM).EQ.'GPSSPROG') --> GPSSPROG
MASSAGER INTERROGATOR
ENCODING(ETYPE(MEM).EQ.'QUESANSW') -->
MASSAGER(QAMODE(MEM)) INTERROGATOR COMPMMSG
ENCODING(ETYPE(MEM).EQ.'CHECKIPR') -->
MASSAGER(-QAMODE(MEM)) INTERROGATOR COMPMMSG
ENCODING --> NULL

```

```

LEXOCLOGY FOR DECODING ENGLISH:
ADJ('MAXIMUM') NOUNPH('CONTENT') --> NOUNPH('SM')
ADJ('AVERAGE') NOUNPH('SR') --> NCUNPH('SR')
ADJ('AVERAGE') NOUNPH('CONTENT') --> NOUNPH('SA')
ADJ('CURRENT') NOUNPH('CONTENT') --> NOUNPH('S')
ADJ('AVERAGE') NOUNPH('TIME') --> NOUNPH('ST')
ADJ('CURRENT') NOUNPH('LINE') --> NOUNPH('Q')
ADJ('AVERAGE') NOUNPH('LINE') --> NOUNPH('QA')
ADJ('MAXIMUM') NOUNPH('LINE') --> NOUNPH('QM')
ADJ('AVERAGE') VERB('WAIT') NOUNPH('TIME') --> NOUNPH('QT')
NOUN('MEAN') NOUN('TRANSIT') NOUNPH('TIME') --> NCUNPH('TB')
NOUNPH('$GPSSSENTY') --> STATHP(@8=PRCODE(SUP(NOUNPH)))
NOUNPH('$ENTITY') NOUNPH('$GPSSSENTY') -->
STATHP(@8=PRCODE(SUP(NOUNPH#2)),@9=IDNO(ENTRY(NOUNPH#1)))
NOUNPH('$STATENTY') --> STATHP(@8=5,@9=IDNO(ENTRY(NOUNPH)))
NOUNPH('$MOBENTY') --> STATHP(@8=3,@9=IDNO(ENTRY(NOUNPH)))
VERB('PRINT') STATHP(NOUNPH('STATIST')) --> PRINTPH(%STATHP)
NOUNPH(-PRM,$GPSSATTR,-OBJ) PREPPH('AT') -->
NOUNPH(OBJ(PREPPH))

```



```

MORPHOLOGY FOR DECODING ENGLISH:
NOUNS('UTILIZA') I O N --> NOUNS('SR')
NOUNS('STATIST') I C --> NCUNS

MORPHOLOGY FOR ENCODING ENGLISH:
NOUNS(CHARS(SUP)) --> NAME(CHARS(SUP(NCUNS)))

DELETE KWDSNT:
SEMIOLOGY FOR DECODING ENGLISH:
KWDSNT(TIMENITY,PROBLEMIRUN|SIMULATE) -->
OK(PRC|TIME(MEM)=TIMENITY(KWDSNT))
KWDSNT(TIMENITY,UNIT) --> OK(TIMUNIT(MEM)=TIMENITY(KWDSNT))
KWDSNT(ENGLISH|STATE|DESCRIBE) -->
ENCODING(ETYPE(MEM)=ENGLISH)
KWDSNT(PRINT,GPSS,-PROGRAM) --> OK(@8=6,SETIND,SIMOUT)
KWDSNT(GPSS|VECTOR) -->
ENCODING(ETYPE(MEM)=GPSSPROG,XVSW(MEM),GPSSSW(MEM),
-GPSS(KWDSNT),-GPSSSW(MEM))

KWDSNT(RESET) -->
OK(IX=1,XV=2,PUTRX,IX=2,XV=1,PUTRX,SIMULT)
KWDSNT(CLEAR) -->
OK(IX=1,XV=3,PUTRX,IX=2,XV=1,PUTRX,SIMULT)
KWDSNT(CONTINUE) -->
OK(IX=1,XV=4,PUTRX,IX=2,XV=1,PUTRX,SIMULT)
KWDSNT(PERFORM|SIMULATE|RUN) -->
KWDSNT(PRINT,CURRENT,EVENTS) --> OK(@8=4,@9=2,SETIND,SIMOUT)
KWDSNT(ASK|QUEST|CN|PROMPT|INQUIRY) -->
ENCODING(ETYPE(MEM)=QUESTANSW)
KWDSNT(OK|OKAY|COMPLETE|DONE|CHECK) -->
ENCODING(ETYPE(MEM)=CHECK|PR) -->
KWDSNT(EXPLAIN|CLARIFY|WHAT|WHICH) -->
ENCODING(ETYPE(MEM)=CHECK|PR) -->
KWDSNT(STOP) -->
OK(-QUESTSW(MEM),-PRED(MEM),-SUCC(MEM),
-CENTY(MEM),-ATTRIB(MEM),-SUPSET(MEM),
-CONDITN(MEM),-QAMODE(MEM))

KWDSNT --> ERROR(MESSAGE2)
PUNCA/ PRINTPH PERIOD -->
OK(%PRINTPH,SETIND,SIMOUT,SPARSE(MEM))

:END OF FILE:

```


LIST OF REFERENCES

1. Minsky, Marvin, ed., Semantic Information Processing, MIT Press, Cambridge, Mass., 1968.
2. Simmons, R. F., "Natural Language Question Answering Systems: 1969", Communications of the ACM, Vol. 13, No. 1, pp. 15-30, January 1970.
3. Lamb, Sydney M., Outline of Stratificational Grammar, Georgetown University Press, Washington, D. C., 1966.
4. Heidorn, George E., Natural Language Inputs to a Simulation Programming System, Technical Report NPS-55HD72101A, Naval Postgraduate School, Monterey, Calif., October 1972.
5. Hansen, Richard D., GES: A Data-Structure-to-GPSS Encoding System, M. S. Thesis, Naval Postgraduate School, Monterey, California, December 1970.
6. McGee, Robert T., The Translation of Data Structure Representations of Simple Queuing Problems into GPSS Programs and English Text, M.S. Thesis, Naval Postgraduate School, Monterey, California, June 1971.
7. Baker, Eldon S., Question-Answer Inputs to a Simulation Program Generating System, M.S. Thesis, Naval Postgraduate School, Monterey, California, June 1971.
8. Hemphill, Frederick H., Computer Verification of the Completeness of a Simulation Problem Description by Natural Language Interaction, M.S. Thesis, Naval Postgraduate School, Monterey, California, December 1971.
9. Williams, Robert J., A GPSS-Like Simulator Callable from a Fortran Program, M.S. Thesis, Naval Postgraduate School, Monterey, California, March 1972.
10. Rickelman, John H., Translation of a Queuing Problem Description Into GPSS-Like Tables, M.S. Thesis, Naval Postgraduate School, Monterey, California, June 1972.
11. International Business Machines Corporation, General Purpose Simulation System/360, User's Manual, 1968.

INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2. Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3. Professor George E. Heidorn, Code 55Hd Department of Operations Research and Administrative Sciences Naval Postgraduate School Monterey, California 93940	5
4. Professor David K. Jefferson, Code 53Jf Department of Mathematics Naval Postgraduate School Monterey, California 93940	1
5. LT Joseph Leon Clapper, USNR Box 19A, R.D.#1 Wernersville, Pennsylvania 19565	1
6. LT Kenneth Stanley Nelson, USN 1426 South Gourley Boise, Idaho 83705	1

Unclassified

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

ORIGINATING ACTIVITY (Corporate author)

Naval Postgraduate School
Monterey, California 93940

2a. REPORT SECURITY CLASSIFICATION

Unclassified

2b. GROUP

REPORT TITLE

Towards an English Language Interactive Simulation System

DESCRIPTIVE NOTES (Type of report and, inclusive dates)

Master's Thesis; December 1972

AUTHOR(S) (First name, middle initial, last name)

Joseph L. Clapper

Kenneth S. Nelson

REPORT DATE

December 1972

7a. TOTAL NO. OF PAGES

111

7b. NO. OF REFS

11

CONTRACT OR GRANT NO.

9a. ORIGINATOR'S REPORT NUMBER(S)

PROJECT NO.

9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

DISTRIBUTION STATEMENT

Approved for public release; distribution unlimited.

SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY

Naval Postgraduate School
Monterey, California 93940

ABSTRACT

Research at the Naval Postgraduate School has led to the development of a system for producing GPSS simulation programs for simple queuing problems through English language dialogue with an IBM 360/67 computer. This thesis describes work done to give the system the capability to actually perform the simulation and report the results through English language dialogue. A complete sample terminal session is included.

KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Digital Computer Simulation						
Simulation Programming						
GPSS						
Computational Linguistics						
Grammar-Rule Language						
Stratificational Grammar						
Natural Language Processing						

141257

Thesis
C4802
c.1

Clapper

Towards an English
language interactive
simulation system.

141257

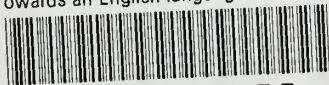
Thesis
C4802
c.1

Clapper

Towards an English
language interactive
simulation system.

thesC4802

Towards an English language interactive



3 2768 002 10247 7
DUDLEY KNOX LIBRARY